Electronic Version
Stylesheet Version v1.1.1

Description

SYSTEM AND METHOD FOR CONTINUOUS STROKE WORD-BASED TEXT INPUT

[0001] Priority Claim

[0002] This application is a Continuation-in-Part application of copending US Patent Application Ser. No. 10/346,366, filed: January 16, 2003, which is hereby incorporated by reference.

FIELD OF THE INVENTION

[0003] The invention relates to a text input system and, more specifically, to a touch screen text input system.

BACKGROUND OF THE INVENTION

[0004] The origin of the modern keyboard as the primary method for inputting text from a human to a machine dates back to early typewriters in the 19th century. As computers were developed, it was a natural evolution to adapt the typewriter keyboard to use as the primary method for inputting text. For a skilled typist, it has remained the fastest way possible to input text into the computer.

[0005] With ongoing efforts to make computers smaller and more portable, the physical keyboard has become one of the most significant limiting factors in just how small a device can become: the physical size of the human finger is not something computer designers could change. As a result, computers for certain portable applications have been designed without a physical keyboard, and use touch-screen based input methods as the primary form of human-computer interface. (This is also the case for some applications where people are physically unable to use a keyboard, such as persons with physical disabilities.)

[0006] Input methods that are designed for portable touch-screen based devices have two primary requirements which frequently conflict with each other. The method of input must be as fast as possible and at the same time the method of input must take as little of the display screen as possible. Unfortunately, as the space taken up for input on the display screen is decreased, it becomes difficult to increase speed without adversely affecting accuracy.

[0007] In spite of a recent surge in the market for pen-based computing devices, most people who must generate text still do so with a standard keyboard. In fact, an entire industry has sprung up that provides portable keyboards for pen-based computers that were designed to be keyboard-less! To date, pen-based computing has not replaced conventional portable laptop computers as was originally forecast, for the simple reason that text input on pen-based computers is too slow. Even the recently introduced "Tablet PC," largely due to the lack of a sufficiently accurate handwriting recognition engine, serves largely as a way to store and retrieve "digital ink" - actual graphic images of handwriting traced on the touch-screen by the user - as opposed to recognizing what was handwritten and converting to computerized text.

[0008] Analogous to one-finger typing, the current state-of-the art for inputting using a virtual keyboard is called "point and tap". A stylus is moved from letter to letter and a "tap" is performed by contacting the screen on the desired key to select it. This results in the need to constantly lift the stylus from the screen and set back down again, significantly slowing input. Cursive handwriting was invented to allow a better (and faster) flow from letter to letter and reduce the number of pen (or quill) lifts. In a similar way, the current invention reduces the number of taps required when inputting using an on-screen keyboard, thus speeding text entry.

[0009] The natural way for a human to create text on anything other than a machine is to “write” it by hand. Accordingly, with the advent of touch-screen computers, it is not surprising that handwriting recognition software was developed to allow a user to enter text by writing on the screen of the computer. However natural, handwriting is slow. Each letter requires several strokes of the stylus, making it very inefficient. Further, with varying handwriting styles, accuracy of this software is still below user-acceptance levels (see MacKenzie, I. S., & Chang, L. (1999), A performance comparison of two handwriting recognizers. Interacting with Computers, 11, 283-297.). As mentioned above, the reliance on the use of “digital ink” by even the “latest and greatest” of computer touch-screen technology, the Tablet PC, provides clear evidence that handwriting recognition is still not good enough to satisfy most users. Furthermore, even if a completely accurate method of handwriting recognition were available, handwriting itself is simply too slow and tiring to provide a satisfactory input method (especially on a touch-screen, which does not provide the same “feel” as a pen on actual paper).

[0010] Some methods make the job easier for the software by requiring the user to handwrite letters in a simplified way (see Goldberg, US Patent Application 20020009227, Unistrokes; or as popularized by Palm Computing in their commercial product titled “Graffiti”). Advantages of this method are that each character is sufficiently unique as to be easily recognized by the software, that each character is drawn as a single stroke, and no virtual keyboard is required on the screen. Disadvantages of this method are it requires the user to learn a new writing method and still requires the stylus to be set-down and lifted once for each individual character (thus slowing text entry).

[0011] Again, it was a natural evolution for the idea of a keyboard to be carried on into the virtual world of the computer display. Auer et al., in US Patent #4,725,694, describe a system wherein one or more images of simulated keyboards are displayed on a touch-sensitive screen of a computer, and in response to the touching of the simulated keys, generate appropriate control signals. In a later refinement of this concept, the image of the keyboard is displayed floating above other applications running on the computer, rather than occupying a dedicated portion of the screen. The user interacts with this “on-screen keyboard" or "virtual keyboard" by either directing a cursor pointer over it, or directly touching the keys via a touch screen using a finger or stylus. On-screen keyboards have been primarily used for devices which lack a standard keyboard, such as certain public information kiosks and personal digital assistants (PDAs), handheld computers that are too small to accommodate a physical keyboard. They are also frequently used by individuals with disabilities that prevent them from using a physical keyboard.

[0012] There are two large shortcomings of on-screen keyboards: first they take up valuable screen space on the computer needed for whatever task is requiring text input. Second, and more importantly, they are slow because the user is forced to tap one letter at a time – effectively reducing the user to input text in a way that is analogous to single finger typing on a regular physical keyboard.

[0013] In an effort to address the slow rate of typing with on-screen keyboards, predictive software was developed which, based on preceding words and on the initial letters typed for the current word, attempts to predict what word is being typed and presents the user with a list of word or phrase choices that they can select as a quicker alternative to completing the word or phrase letter by letter. Due to the need to divert attention from the task at hand (typing a word) in order to scan the prediction list and determine whether the intended word has been offered as a choice, this “Word Prediction” scheme offers only a marginal increase in speed of text entry, if any (depending on the user).

[0014] As the size of the on-screen keyboard is reduced beyond a certain point, the speed of text entry is sharply reduced. This is due to the requirement for increased accuracy and dexterity in hitting the smaller targets. Various schemes have been developed to minimize the size of the keyboard, yet still maintain accuracy without unduly sacrificing speed of entry.

[0015] Grover et al., in US Patent #5,818,437, describe a system that reduces the number of distinct keys required by assigning multiple letters on each key. This allows, for a given size of keyboard, relatively larger individual keys which are therefore easier to hit accurately, thus allowing the user to type more quickly. Text entry in this system is word-based, so that disambiguation software uses a database of words to analyze each sequence of keystrokes and determine the most likely word (or words) corresponding to the sequence, and consequently determine which letter was actually intended by each ambiguous keystroke. While the system of Grover et al. makes it easier to hit an intended key by virtue of reducing the total number of keys and enlarging the individual keys, when implemented on a touch-screen device, it still requires the user to lift the stylus and set it down for each letter entered, significantly slowing down text entry. Furthermore, this approach requires the user to a very unfamiliar keyboard layout in which completely unrelated letters are grouped together on a single key. Even when letters are grouped according to an "alphabetic" layout (as on the keys of cellular telephone), compared to the standard "QWERTY" keyboard, the arrangement is unfamiliar for the majority of individuals and further slows the text entry process.

[0016] Lee, in US Patent #6,292,179, describes another system that reduces the number of distinct keys required on a touch screen keyboard by assigning multiple letters on each key, and determining which letter associated with a contacted key is intended by determining the direction in which the stylus is moved after contacting a key. Each letter associated with a key is further associated with a range of directions in which the point of contact can be moved.

Lee's method also allows each key of a given keyboard to be relatively larger, and therefore easier to initially contact, since multiple letters are combined into a single key and a smaller total number of keys is therefore required. However, the user is still required to tap once for each desired letter, and is further required to move the point of contact in a particular direction before lifting the stylus and breaking contact with the screen.

[0017] Kaehler, in US Patent #5,128,672, describes another system designed to reduce the number of distinct keys that are required for a touch screen keyboard by displaying at any given time only a subset of the total set of characters that can be entered. The system attempts to determine the subset of characters comprising the most likely next characters to be entered, based on the previous character entered or the positioning of the text insertion point. When the desired character does not appear, the user must manually switch to a different keyboard to locate and enter the desired character. The large number of different (and constantly changing) partial keyboards would tend to make this a slow and frustrating input method for the majority of users.

[0018] Vargas, in US Patent #5,748,512, attempted to reduce the need for accuracy on a touch screen keyboard (and therefore increase speed) by considering two adjacent keys as possible candidates when a key is not activated in its central area. Based on the actual location at which the keyboard was contacted relative to the three keys, combined with the statistical analysis of preceding characters in the word being entered (if any), and optionally also using information from a word prediction engine, the system determines the most likely of the three possible candidate characters and displays it as the character to be input in response to the activation. However, since each character, once input, forms the basis for the prediction of subsequent characters, when a character prediction is incorrect, it must be corrected by the user before he can proceed to type the next character. In the system as described, this is done by maintaining contact with the keyboard during an activation, observing whether the predicted character is correct, and if necessary, sliding the point of contact in the direction of the actual intended letter. The advantage of this invention, as described, is that it enables the user to use something blunt (such as a fingertip, rather than a stylus) to activate keyboard keys that are actually smaller than the instrument of activation. However, the nature of the feedback provided to the user and the consequent need to observe the result of each keystroke and correct it before moving on to the next keystroke, creates a system that generally would slow the rate of text input considerably.

[0019] Robinson et al., in international patent publication WO 00/74240 A1, describe a text input system for touch-screen devices that includes a keyboard with an auto-correcting region that includes the set of keys that are associated with letters. The advantage of the system is that, for words that are included in the system database, the user does not need to contact within the region of the key associated with desired letter, but instead need only tap in the neighborhood of the key. The user taps the keyboard once for each letter in the word being entered, and the system records the location of each contact. The sequence of contacts is then compared with the key locations associated with words in the database, and the most likely one or more matching words are presented to the user for selection. This system is a significant improvement over previous approaches in that it enables the user to type much more quickly on a small keyboard because it is no longer necessary to precisely contact within the region of each intended key. However, for each key activation, the user still needs to touch down on the screen with control, targeting the intended key, then lift the stylus from the screen and move to target the next key. The additional movements of lifting and setting down the stylus for each letter, combined with the additional effort required to control the relocation of the stylus when it is not in contact with the screen, result in significantly slowing down the input process compared to the system of the present invention.

[0020] Another factor in slowing text entry on touch-screen keyboards was the time it takes to lift the stylus from the screen and then bring it back down between each key selection (“tapping”). United States Patent 5,574,482 (Niemeier) discloses a method for data input on a touch sensitive screen. The Niemeier patent teaches having what are described as computer generated "temporary" adjacent keys which can be made to appear on top of a standard keyboard layout. When the user touches a key, selecting a first letter, one or more temporary keys are displayed adjacent to the contacted key as long as the initial contact is maintained. A second letter (or group of letters) that is displayed on an adjacent temporary key can then be selected by making what is described as a "wiping" motion in which one's finger or a stylus is slid from the first selected key to the adjacent temporary key. This teaching enables two (or more) adjacent letters to be input without lifting the stylus from the screen, approximately cutting in half the number of times the stylus needs to be lifted from the touch screen. The "temporary" adjacent keys create artificial groupings of the most probable letters to provide more opportunity for "wiping" input.

[0021] However, the method described by Niemeier has several significant drawbacks. One is that the user needs to either memorize 26 new “temporary” sub-keyboards that appear when each of the 26 letter keys are contacted (creating a significant learning curve in order to use the system effectively), or else the user needs to pause to observe whether and where the next desired key may have appeared on a temporary key, which would likely negate any speed advantage that the “tap and slide” method would provide. The situation becomes worse when, in order to increase the likelihood that the desired letter will appear, a dictionary or other database is used to alter the temporary keys that are displayed for each key based on the context of the previous letters that that were selected prior to activating the key. Furthermore, as described above, the system is limited to sliding to at most one additional key beyond the initially selected key, unless the subsequent letter also happens to appear on a key adjacent to the selected temporary key. Also, the number of temporary keys that can be displayed (and therefore selected) is limited to the number that can be displayed around the circumference of a single key (six for standard key arrangements as proposed by Niemeier, up to a maximum of eight). Furthermore, since the temporary keys are only displayed while the stylus (or finger) is actually in contact with the screen, a significant number of the keys which might be displayed would likely be partially or totally obscured by the stylus and hand of the user. Also, the system proposed by Niemeier includes a significant amount of space in between active keys (used for capitalization and spacing), reducing the size of the actual key targets, each of which must be contacted in the interior of the defined key area in order to effect an activation.

[0022] Van Kleeck, in US Patent #6,008,799, describes a somewhat similar system, wherein the “temporary keys” are never actually displayed, but where each of the four vowels “o”, “e”, “i” and “a” are implicitly associated with each letter key, and can be appended by tapping a character button and dragging the pen in either the north, south, east or west direction. While this approach is easier to learn than Niemeier's (since only the four vowels and their associated directions must be learned), and it is not affected by potential visual interference by the user's hand (since no temporary keys are actually displayed), it is limited to only being able to add one of the four particular letters following a key selection, and thus does not offer a very significant improvement over a standard keyboard.

[0023] Perlin describes a method in US Patent # 6,031,525 where the stylus is never lifted from the touch screen, but rather directed from a middle neutral point (a “resting zone” in Perlin's terminology) to one of a number of surrounding zones, each of which contains a plurality of letters. The user is required to contact the screen in the resting zone and then perform a continuous sliding motion which, for each letter to be entered, goes from the resting zone out into the adjacent zone that contains the desired character, and then indicates which character contained in the zone is intended by either returning directly to the resting zone or by first traveling to a different zone before returning to the resting zone. Thus, the desired character is indicated by the combination of the zone first entered from the resting zone, combined with the zone from which the resting zone is again re-entered. As a result, the user is required to make a minimum of 2 and more commonly 3 strokes of the stylus to indicate each desired letter. This method allows for a smaller keyboard, but requires multiple strokes per letter which dramatically reduces the speed of text entry.

[0024] Another common approach in speeding text entry is to modify the layout of characters on the keys of an on-screen keyboard (see www.fitaly.com). Common letters are placed near the center of the onscreen keyboard, and common letter pairs are made adjacent. In this way, stylus movement between letters is minimized, thus speeding text entry. The disadvantage of this method is it requires the user to learn a new keyboard layout, and again, the user is still required to lift the stylus between each key activation, essentially reducing the user to a one-finger typist.

[0025] Each of the above methods has the potential to speed text entry by some degree and/or decrease the amount of on-screen real estate required for input. However, text entry via existing on-screen keyboards and handwriting recognition techniques are still generally slower than using a physical keyboard. Handwriting itself is simply too slow, and the recognition accuracy is still not up to acceptable standards. All of the keyboard-based methods described above require one or more separate and distinct actions for the input of each individual letter. Most of the proposed keyboard systems are based on tapping some type of key, so that generating a single character requires that the stylus be appropriately positioned, the on-screen keyboard be contacted, and the stylus lifted from the screen again before proceeding to input the next character.

[0026] The fundamental problem is that, particularly for devices that must utilize touch-screen keyboards that are reduced in size, the additional actions of lifting the stylus and bringing it back into contact with the screen in a controlled manner considerably slow down the input process. Other methods which have been proposed to reduce the number of times the stylus must be lifted from the screen (such as Niemeier and Perlin) still fail to provide a method that can significantly speed text entry. Niemeier's proposed system adds too much complexity by requiring the user to react to a constantly changing keyboard layout, and limits the number of characters that can be selected following an initial character selection. Perlin's approach fails because it requires the user to execute too many distinct stylus movements to enter each letter.

[0027] Cirrin (Circular Input), designed by Mankoff and Abowd [Mankoff, J. and G.D. Abowd. Cirrin: a word-level unistroke keyboard for pen input. Proc. ACM UIST, Tech. Note. 1998. p. 213 - 214.], presents a keyboard word input system. Cirrin operates by tracing a path through letters arranged in a circle. The user enters a word by moving the stylus through the desired letters in the intended sequence. Although Cirrin is designed to input text one word at a time - the pen contacts the screen prior to passing through the first letter and lifts up after the last letter of each word - Cirrin does not perform any kind of recognition. This is an advantage in the sense that processing is greatly simplified and no database of words is required by the system. However, it is this limitation that requires the circular configuration of the keyboard. Any letter that is contacted by the stylus is added to the word being output, so the key configuration must be such that the stylus can travel between any two arbitrary letters without contacting another letter. This tends to significantly increase the average distance the stylus must travel to enter a word. It also means that the user must be careful and precise enough that every intended letter is explicitly contacted by the stylus, and conversely, that no other unintended letter is contacted. Although Cirrin attempts to optimize pen movement by arranging the most commonly co-occurring letters closer to each other, this does not greatly reduce the negative impact of the above limitations.

[0028] One other approach has been presented in which a path is traced on a virtual keyboard displayed on a touch-screen in order to enter a word. While the approach presented begins to address some of the concerns mentioned with respect to the other approaches discussed above, there are still a number of significant limitations to the approach. One is that the user is required to make some kind of distinctive indication at every key associated with a word being input. When three letters are co-linear on the virtual keyboard (as in a presented example of the word "pity" in which the first three letters occur in right-to-left sequence in the top row of the keyboard), the user is still required to move the stylus in a distinctive fashion at each letter. In this case, the illustrated example shows a stylus trace that "zig-zags" above and below the top row of keys at the position of the “i” key to indicate the user's intention to input the letter “i”. Alternatively, the user can change the speed or pressure of the stylus motion or pause over the intended letter “i”. Any of these variations in stylus movement require additional time and thus slow the overall rate of text input that is possible. The approach also proposes that, all characters in the immediate vicinity of a “selection” (this term is undefined, but appears to refer to the location determined for a change in direction or pause in stylus movement) are considered as possible candidates. However, no definition is presented as to what constitutes a “change in direction.” In the approach presented, the actual extent of the “vicinity” of a selection is determined by the user choosing a fixed radius that extends from the location of each selection and within which all characters are considered as candidates for the letter intended by the distinct movement. The approach then proposes that statistical models are used to determine which of these candidate characters is most likely intended. With sufficient accuracy by the user in the locations of the majority of distinct movements (such that the intended letter is unambiguously deduced), this approach could be used to attempt to resolve ambiguity for a relatively small number of the letters of a given word. However, this has two major limitations. One is that in many cases, the letter determined by statistical modeling would not in fact be the intended letter, forcing the user to edit or re-enter the word. The second is the simple fact that the majority of distinctive movements of the stylus would have to be quite accurate in order to provide a context with enough information to resolve the ambiguity of the remaining letters in each word. This requirement for significant accuracy on the part of the user drastically lowers the speed with which the user can then enter each word, since faster stylus movement inevitably results in lowered accuracy of movement.

[0029] Zhai [Zhai, S., Kristensson, P-O., "Shorthand Writing on Stylus Keyboard" in *Proc of CHI 2003 - ACM Conference on Human Factors in Computing Systems*. pp 97-104; and Zhai, S., Kristensson, P-O, Smith, B.A., In search of effective text interfaces for off the desktop computing, to appear in *Interacting with Computers*, Vol.16.] proposes a system that also enables a user to enter a word by tracing a path through the letters of the word on a virtual keyboard displayed on a touchscreen. Zhai calls his system "Shorthand Aided Rapid Keyboarding," designated by the acronym

SHARK. In contrast to the present invention, Zhai's approach is focused on the goal of enabling the user to memorize the pattern corresponding to any word they wish to enter so that the pattern can then be entered as a "shorthand" gesture that the user performs *independently of any displayed keyboard*. One of Zhai's guiding principles is to create a system that reduces the amount of visual attention required of the user in order to allow the user to attend to other tasks or information on the display while inputting text. Zhai asserts: that requiring the user to "To precisely cross all letters defining a word would require just as much, if not more, visual attention as serially tapping all the letters. Furthermore, the time to perform such visually guided steering cannot be expected to be any shorter than tapping. Therefore, for gesturing to be effective, patterns must be recognized at least partially independent of scale and location. As long as the user produces a pattern that matches the shape of the prototype of a word, the system should recognize and type the corresponding word for the user. If so, the users could produce these patterns with much less visual attention and presumably greater ease and comfort. Scale and location independency is the first principle in our current work." As a result, Zhai proposes a recognizer based on a standard elastic matching algorithm, “which computes the minimum distance between two sets of points by dynamic programming. One set of points is from the shape that a user produces (a unknown shape). The other is from a prototype – the ideal shape defined by the letter key positions of a word. After preprocessing, filtering and normalization in scale, the distance between the unknown shape and the prototypes are computed by elastic matching. The word corresponding to the prototype that has the shortest distance to the sample is returned as the recognized word.” This type of pattern recognition emphasizes the degree of proportion between the user's gesture and the prototype of a potentially matching word.

[0030] This approach is a natural result of Zhai's emphasis on scale and location independence as the first principle in his work. However, while Zhai's approach allows the user's gesture to be performed independently of the keyboard, and without regard to the size or position of the gesture, the emphasis on proportionality means that, even if the gesture *is* in fact performed on the keyboard (so that the positions of the keys form a visual template that the user can follow in entering the gesture), then the gesture must be entered with a relatively small amount of error (or, even more unlikely, with roughly the same error in the same direction from each key, or still less likely, with an error that is approximately proportional either “outward” from the center of the gesture or “inward” toward its center). The term “sokgraph,” which appears in Zhai's terminology, will be used here as the term for an input gesture performed in a size and location independent manner. For example, on a Qwerty keyboard layout, the words “put”, “pit”, “pot” and “pout” collide by virtue of the fact that they all have identical input paths consisting of a straight line path from the “p” to the “t” key. Zhai's approach is unable to distinguish between such words, except by means of adding an additional required stroke on a pop-up “pie menu” to designate which of the otherwise indistinguishable words is intended by the user. Zhai's keyboard configuration is designed to result in fewer such collisions but at the cost of forcing the user to adapt to a new and unfamiliar keyboard configuration. With Zhai's approach that requires that sokgraphs be recognized in a size and location independent fashion, the number of such collisions necessarily increases (e.g. to/ top/can/aim/am/an/less all have a sokgraph consisting of a straight, horizontal left to right stroke, and is/my/no/at all consist of a straight diagonal upper-left to lower-right stroke). Despite Zhai's unusual “ATOMIK” keyboard configuration, there are still pairs of words whose input paths cannot be distinguished even when the principle of scale and location independence is ignored (e.g. lip/lisp; cosmic/comic; bald/ballad; care/carte; aim/am; claim/clam; dept/depot; pales/pals; etc.).

[0031] Zhai cites his second guiding principle as “efficiency” which, Zhai argues, motivates the use of the “ATOMIK” keyboard layout rather than the standard QWERTY configuration. Zhai argues that the QWERTY layout “would involve frequent left-right zigzag strokes, because the commonly used consecutive keys are deliberately arranged on the opposite sides of QWERTY” whereas the ATOMIK layout was designed such that “in any given scale, the average word gesturing length on ATOMIK is also minimized.” The ATOMIK layout also “maximizes, without sacrificing the first two features, the letter connectivity of the most common words." Since common words tend to be short, and since there are at most six ways in which letters can be connected, this approach tends to increase the number of pairs of common words whose sokgraphs cannot be distinguished on a scale- and location-independent basis. An important design goal of the present invention is to enable the user to use gestures to input all words entered with the input system. The two factors of minimizing gesture length and maximizing letter connectivity combine to make it difficult, if not impossible, to reliably distinguish between gestures in system designed to enable gestural input of a large vocabulary (over 60,000 words in current implementations of the present invention).

[0032] Zhai's acceptance of such limitations is consistent with interpretation and application of Zipf's law to the design of his proposed approach. Zipf's law is the simple fact that, in virtually any language, a relatively small number of frequently used words account for a disproportionately large percentage of the words used in a corpus representative of an “average” user. Zhai takes this fact as the basis of his rationale that the system is effective even when the user learns (and therefore uses) only a limited number of gestures corresponding to the more frequent (and therefore, statistically, significantly shorter) words. Zhai even explicitly describes this as the underlying principle of “duality” in his approach: “A shorthand system defined on a stylus keyboard, however, does not have to contain a complete or even a large set of words, because one can use both tapping and shorthand gesturing. For familiar words whose patterns are well remembered, the user can write their shorthand. For the less familiar, one can use stylus tapping." Thus, Zhai's approach is designed to accommodate gesturing for only a limited portion of the words to be input. In contrast, the present invention allows gestural input of virtually every word, with the very rare exception of a word not yet present in the lexical database of the system, which therefore must be tapped in the first time it is entered in order to add it to the database, so that a gesture can be used thereafter to input the previously unknown word.

[0033] Zhai enumerates a fifth fundamental principle underlying his approach is the idea that there should (or must) be a gradual transition from tapping to shorthand. This is exemplified in his name for the SHARK system – "Shorthand *Aided* Rapid Keyboarding." Zhai does not even contemplate a system in which virtually every word is input by means of a gesture. Zhai states explicitly: "a user's repertoire of shorthand symbols can be gradually expanded with practice. Gesturing and tapping a word with SHARK share a common movement pattern, which may facilitate skill transfer between the two modes. For a novice user, visually guided tapping is easier. When a word is tapped enough times, the user may switch to the more fluid "expert" mode of shorthand gesturing. If a shorthand gesture is forgotten, one can fall back to taping [sic], which reinforces the pattern and pushes the user back to expert mode." Zhai's requirement for scale and location independent gesturing (which significantly affects his proposed approach to gesture recognition), and the consequent need for the user to memorize each gesture to be used, necessitates a continued dependence on tapping in Zhai's system.

[0034] Thus, Zhai consistently and emphatically teaches away from the concept that gesturing can or should be performed with respect to the keys displayed on a virtual keyboard. As quoted above, Zhai asserts that "such visually guided steering cannot be expected to be any shorter than tapping" based on theoretical calculations in which "the law of steering [1] and Fitts' law of pointing [are used] to quantitatively analyze the difference between tapping and tracing." However, these calculations are based on the assumption (which, as shown above, follows from the fundamental guiding principles adopted by Zhai) that the user must “precisely cross all letters defining a word” when entering the input path for a word. From these assumptions, Zhai concludes that gesturing neither can nor should be used to enter every word when inputting text.

[0035] Thus there is a need for a system which can accommodate the use of gesturing for all text to be input (with the possible exception of the one-time tapping entry of a word not previously known to the system). As it is impractical to expect that users will memorize the gestures for all of the words in their vocabulary, there is also a need for an approach which does not require this. Finally, to accommodate greater speed in the entry of gestures, there is a need to allow the user to be less precise in entering gestures without sacrificing the ability to consistently interpret the user's gestures in such a way that they are correctly matched with the intended word.

[0036] As described below, the various aspects of the present invention combine to yield an unexpected result - that visually guided gesturing with reference to a displayed virtual keyboard is far faster than tapping, and can be used to enter words from a vocabulary that is large enough (for example, over 60,000 words in one implementation of a system for the English language) that the user only needs to resort to tapping to enter an unusual name or word the first time that word is entered using the system. This initial entry of a new word can reasonably be viewed as a process that is outside the normal text entry method, since once such a word has been tapped in it is then known to the system and can thereafter be entered using gesturing. The present invention, however, also includes aspects with respect to the tapping of words that improve upon anything known in the prior art. And, as will be shown in the following description, the present system allows the user to easily enter distinct words that “collide” because they otherwise map to the same basic gesture, or input path.

SUMMARY OF THE INVENTION

[0037] The method of the present invention is distinguished by the fact that each word is input using one continuous motion in contacting a virtual keyboard on a screen, significantly reducing the number of controlled movements that must be executed to input each word. This significantly increases text entry speed. The present invention uses word-level analysis to match the input pattern traced out on the keyboard with the most likely word or words in the system's database. The user is presented with a list of the identified matching words, and can either accept the de-fault choice (the word identified by the system as being the most likely match), or select one of the alternate word choices or request the system to display further poten-tially matching words if the desired word does not appear in the list. When the system determines that the likelihood that the default choice is in fact the user's intended word is sufficiently greater than the likelihood of the second-most-likely word, then the default word is output without displaying a list of alternate word choices. Other aspects of the system provide feedback to the user in such a way as to minimize the necessity to distract the user from the task of composing and entering text.

[0038] When a word is generated for output immediately follow-ing a previously output word, or immediately following one of a set of punctuation characters that are normally followed by a space, the system automatically outputs a space prior to outputting the generated word, eliminating the need to perform any additional action for the vast ma-jority of space characters typed.

[0039] The system also allows words not yet present in the sys-tem database to be entered once by tapping, at which time they are added automatically to the database such that they can subsequently be input using the continuous stroke method.

[0040] The method of the present invention has a number of very significant advantages over prior systems such as those disclosed by Niemeier and Perlin. One is that the keyboard displayed by the system stays constant, with the same letters always appearing in the same positions. This is in contrast with the system proposed by Niemeier, in which differing sets of temporary keys appear each time the screen is contacted, forcing the user to observe and react to a dynamically changing layout. In addition, this allows the method of the present invention to be used with a static keyboard that is imprinted on a less-costly touch-sensitive membrane rather than a touch-sensitive dynamic display. A second advantage is its great economy of movement. The stylus need only be brought into contact at the start of each word, and lifted at the end of each word. As will be described below, in the vast majority of cases no additional action whatsoever is required to correctly generate a space between each generated word. Furthermore, in contrast to the system of Perlin, the stylus need only be moved directly from letter to letter of the word being input, rather than having to perform two to three distinct strokes for each letter. A third advantage is that the individual letters of a word need not be contacted precisely with the stylus. The stylus need only be brought into the neighborhood of each letter in sequence, and need not necessarily pass directly through the region defined for the key associated with the letter. The system allows the user to be proportionately less precise (and consequently, to have the potential to move the stylus more quickly) when inputting a word with a relatively higher frequency of use. This greatly increases the overall speed of input, since it enables to user to input most quickly those words that tend to be used most often.

[0041] A further advantage is obtained for the present invention through the insight that, due to the effect of practice, a user of the system will naturally tend to trace out paths more quickly for words with which the user is more familiar (which tend to be generally higher-frequency words). Conversely, when the user is entering a word that is unfamiliar, seldom used, or of whose spelling the user is unsure, the input path will tend to be traced out at a slower speed. This relationship can be used to change a number of the parameters used in carrying out the analysis of the input path itself and the subsequent comparison of the analyzed path with potentially matching words in the database. The effect of stylus speed on the analysis of the input path can be at either or both of two levels. In one aspect, the average stylus speed along the entire current input path is determined and compared to the average stylus speed calculated for a determined number of the previously entered input paths. The result of the comparison (for example, the ratio of the two average speeds) is then used to adjust the values of various analysis parameters that apply to the entire input path, as detailed below in some exemplary embodiments. In another aspect, the speed of the stylus as it passes through a determined neighborhood of a point on the input path is compared to the average stylus speed along the entire current input path, and the result of the comparison is then used to adjust the values of various analysis parameters that apply to the local point or segment of the input path.

[0042] The present invention provides a keyboard text entry system for devices with touch-sensitive input panels or touch-sensitive display screens. More specifically, the present invention provides a system and method that enables the user to enter text word-by-word using a keyboard displayed or printed on a touch-sensitive screen, wherein contact with the surface of the display generates input signals to the system corresponding to the location of contact. The user enters a word by contacting the screen and tracing out a continuous pattern that passes within a threshold distance of each letter of the word in sequence, and breaking contact with the screen when the last letter of the word is reached.

[0043] The present invention includes a number of aspects whereby the magnitude of such letter-to-path threshold distances are adjusted or weighted as a function of one or more of a number of factors, including but not limited to pre-defined threshold values, the specific characteristics of the input path being analyzed, and the particular history of input paths entered by the user. In general, increasing the magnitude of such distance thresholds enables the system to match a less precisely entered input path with the user's intended word.

[0044] In another aspect, the keyboard is displayed on a touch-sensitive display screen (hereinafter referred to as a touch-screen) and the user contacts the display by means of a stylus. It should be understood, however, that the system can be applied to any system where the user can trace out a continuous path on a displayed keyboard, for example, a touch-sensitive screen that is contacted with the user's finger, or even a standard computer display monitor (not a touch-sensitive screen) where the point of “contact” is the location of an on-screen cursor whose location on the display is controlled by a mouse (or equivalent control device) and where the acts of “contacting” and “breaking contact with” the screen are indicated by closing and opening a switch (or performing some other equivalent control action, such as “dwelling” near a location without moving the mouse for a period of time longer than a selected threshold). The operation of the system will be described with reference to an aspect comprising a touch-screen contacted by a stylus, but this should not be construed as limiting the scope of the invention, but simply as a means to provide illustrations of some of the present aspects of this method. The method simply processes a user-generated stream of location data, which begin at a known point (in the touch-screen model, the point where the stylus first contacts the screen), proceed through a sequence of two-dimensional data point locations which have a known relationship to a defined virtual keyboard area in the same two-dimensional space (in the touch-screen model, the sequence of points detected where the stylus is moved while in contact with the screen), and which end at a known point (in the touch-screen model, the point where the stylus last contacts the screen). In certain embodiments, the times when the data points are generated, or equivalently for these embodiments, the time intervals between the data points, are also processed, and can also be provided by various equivalent technologies.

[0045] The system of the present invention allows the user to input a word of text without having to set the stylus down on the screen to contact an intended letter and then lift the stylus from the screen again before contacting the next letter - i.e., without having to "tap" the key associated with each letter. This enables the user to input text much more quickly, since the extraneous movements of lifting and setting down the stylus need not be performed, and since maintaining contact between the stylus and the screen makes it easier in general to maintain more precise control over the location of contact by helping to stabilize the relationship between the stylus and the screen. Furthermore, in general it allows the displayed keyboard as a whole to be significantly reduced in size, since the path traced out by the user need not precisely contact each letter of the intended word. To the extent that the keyboard is not significantly reduced in size, speed of entry tends to be able to be correspondingly increased. To the extent that the keyboard is large enough to occupy a significant portion of the user's peripheral vision, an experienced user can effectively enter words without devoting visual attention exclusively to the keyboard.

[0046] The path traced out on the touch-screen by the user and recorded by the system for analysis is referred to as the input pattern. As the user traces out an input pattern on the touch-screen, the system records the sequence of points of contact detected by the touch-screen controller hardware. As the input pattern is recorded, it is processed by an input pattern analysis component. The input pattern analysis component extracts the data needed by the pattern matching component, which compares the extracted data with words in a database to identify a list of one or more words determined to be the most likely matching candidate words, and to determine a metric value for each compared word that measures how closely the input pattern matches the word (hereinafter, the “scoring metric”). In one embodiment, when the difference between the scoring metric of the most closely matching word and that of the second-most-closely matching word exceeds a determined threshold, then the most closely matching word is automatically accepted as the textual interpretation of the input pattern and is generated as the output of the system in response to the user input pattern. When the difference between the scoring metric values of the two most-closely-matching words falls below the determined threshold, then two or more of these identified words are presented to the user for selection, wherein the most closely matching word is presented as a default word choice, and a selected word is added to the text being entered by the user. In another embodiment, when two or more identified words are presented to the user for selection and the user determines that the default word choice is the intended word, and no explicit selection action is performed (for example, the user simply begins to enter a new input pattern), then the default word is automatically accepted as the textual interpretation of the input pattern and is generated as the output of the system in response to the user input pattern.

[0047] In another aspect, the input pattern analysis component analyzes the input path to identify “inflection points” where a significant change in the input path is detected, such as, for example, a rapid change in direction. Such inflection points can be detected and extracted through a variety of analysis methods, as explained below. In one alternate aspect, the methods by which various inflection points are detected are associated with varying levels of confidence that they should in fact be associated with the location of a key associated with a letter of the word being input, and additionally with various methods of weighting the distance of a key from the input path, wherein, for example, the horizontal and vertical distances of a key from the path may be differentially weighted. Thus, when the input pattern analysis determines that an inflection point is of a type wherein its location can be accurately determined and wherein there is a very high likelihood that the inflection point in fact corresponds to a letter of the word being input (barring a gross user input error, which is handles in a separate embodiment), then potentially matching words in the database will be deemed more likely matches the closer the corresponding letter of the word is to the determined location. When an inflection point is determined to be of a less reliable type, then the likelihood determined for potentially matching words will be less affected by the distance of the corresponding letter from the inflection point.

[0048] In another aspect, the text input system includes a keyboard implemented on a touch-sensitive display screen, where the location of each displayed text character key ("letter-key") is defined by the screen coordinates of the center of the key, which is the location used when determining the distance of the letter associated with the key from any point on the input pattern. The system also includes a record of the input pattern that includes the coordinate locations of a sequence of points of contact detected from a first location of contact through a final location at which the stylus was lifted from the screen, and a routine to analyze the input pattern to determine the locations associated with one or more inflection points of one or more types, and to calculate the distance between each determined inflection point location and the locations associated with letter-keys. Also, the system includes a database of words that can be entered using the system, a routine to determine a numerical scoring metric for each of one or more words to evaluate which words most closely match the input path by matching in sequence each of the letters of a word with the determined locations of inflection points and/or points along the intervening path segments and calculating a scoring metric that is a function of the distances of the letter-keys of the word from the path points with which they are associated and the nature of the association. Another routine is included to determine whether the scoring metric values for the two most-closely-matching words differ by more than a threshold amount, and a means is included to allow the user to select the desired word from the set of words determined to be the most likely matching candidates when the words differ by less than said threshold amount. In another aspect, the distance of a letter-key from a point on the input pattern is defined as zero when the point on the input pattern lies within the defined boundary of the key, and otherwise is defined as the distance from the point on the input pattern to the nearest point on the defined boundary of the key. Hereinafter, the expression "letter-key of a word" refers to the letter key with which a character in the spelling of a word is associated.

[0049] The term "letter" in the context of the present invention is to be understood to include any character that appears in the spelling of one or more words of the database. The term "word" in the context of the present invention is to be understood to include any sequence of characters that is stored in the database that can be matched against an input path and output as text. Thus, for example, if the word “can't” is among the words in the database of the system, it is possible to enter the word by tracing out a path that starts near the letter “c,” passes through or near the letters “a” and “n,” then through or near the key associated with the apostrophe, and ends near the letter “t.” Similarly, hyphenated words, alphanumeric words, and other words containing special characters can all be included in the database and entered as text using the system of the present invention, providing that each of the characters used in the database is associated with at least one key on the keyboard.

[0050] In another aspect, the input pattern analysis component determines the sequence of first and second order differences (corresponding to rates of change) of the x- and y-coordinates of the sequence of points in the input pattern. The ratio of the x and y first order differences corresponds to the “slope” of the input pattern at each point, such that the second order difference corresponds to the rate of change in the slope. A second order difference hovering near zero corresponds to a segment of the input pattern which is a relatively straight line. A small, relatively constant second order difference indicates a constant rate of change in the slope corresponding to a segment of the input pattern which has a slight, constant curvature. A sharp peak or rapid change in the second order difference corresponds to a relatively sharp change in direction of the input pattern. In another aspect, since the magnitude of the first and second order differences is also a function of the frequency with which contact location data points are sampled and collected by the operating system as well as the speed at which the user is moving the point of contact, the first and second order differences at each point along the input path are calculated with respect to two points at a fixed distance preceding and following the given point along the input path. In another aspect, to simplify computational requirements, this fixed distance is approximated by a fixed sum of the absolute magnitude of the x- and y- first order differences.

[0051] In yet another aspect, when the system detects that the input path has crossed over itself in a loop (as in the entry of a small circle in the entry of a DOUBLE_LETTER gesture, as defined below), the magnitude of the fixed distance used is reduced to approximately the radius of the loop, and the magnitude of the second order difference calculated is scaled according to the ratio of the standard fixed distance to the reduced fixed distance used.

[0052] In another aspect, the input pattern analysis component identifies different types of inflection points in the input pattern. For example, PEN_DOWN, the location where the stylus first makes contact with the touch-screen, PEN_UP, the location where the stylus breaks contact with the touch-screen, ANGLE_THRESHOLD, a location where there is a significant change in the direction of the input path, and TAP, a location where the stylus is more or less immediately lifted after contacting the screen, corresponding to a case of a one-letter word or the selection of a single function key. In one embodiment, the ANGLE_THRESHOLD inflection point is identified at a location where the sum of the absolute magnitudes of the x and y second order differences reaches a local maximum, having exceeded a pre-determined minimum threshold.

[0053] In another aspect, an additional type of inflection point is defined which corresponds to a location where the stylus paused in the trajectory of the input path for more than a determined threshold time interval (a PAUSE inflection point). In another aspect, more than one type of PAUSE inflection point is defined based on two or more distinct threshold time intervals. In another aspect, an additional type of inflection point is defined which corresponds to an ANGLE_THRESHOLD inflection point where it is also determined that the stylus paused in the trajectory of the input path for more than a determined threshold time interval within a determined distance along the input path from the location determined for the ANGLE_THRESHOLD inflection point (a PAUSE_ANGLE inflection point).

[0054] In another aspect, an additional type of inflection point is defined which corresponds to a predetermined type of stylus movement, or gesture, that the user is required to execute to indicate entry of a double letter (DOUBLE_LETTER). To illustrate this, consider the possible entry of the words "feel" and "fell." To enter either word, the user would contact the touch-screen at or near the key associated with the letter "f," pass through or near the key associated with the letter "e," move the stylus to or near the key associated with the letter "l," and lift the stylus from the screen. Without such a DOUBLE_LETTER inflection point type, these words could not be distinguished from each other with respect to an appropriately entered input pattern, forcing the user always to explicitly select one of the two word forms, since only one can be displayed as a default (automatically accepted) word choice. When a DOUBLE_LETTER inflection point type is included, a distinguishable movement would be performed at or near the key associated with the letter "e" in the case of the word "feel," but at or near the key associated with the letter "l" in the case of the word "fell," enabling the system to distinguish between the input patterns for the two words. In another aspect, the movement associated with a DOUBLE_LETTER inflection point type is a relatively small circular motion of the stylus at or near the location of the key associated with the letter to be doubled. The location of a DOUBLE_LETTER inflection point is defined as the center of the relatively small circular gesture traced by the user. In another aspect, each successive additional repetition of the movement associated with a DOUBLE_LETTER inflection point denotes an additional occurrence of the letter in the word being entered. For example, the word "AAA" would be entered by contacting the screen at or near the key associated with the letter "a," executing two small circles with the stylus, and lifting the stylus from the touch-screen.

[0055] In one aspect, the scoring metric function is calculated as the sum of the distances from the identified inflection points to the keys associated with the letters with which the inflection points are matched, plus the distances to any additional unmatched letters where each is measured from the nearest point along the input pattern as described above. In another aspect, the scoring metric function is calculated as the sum of the squares of these distances. In another aspect, prior to calculating the sum, a weighting function is applied to each distance (or squared distance) where the weight applied to each distance is determined according to the type of inflection point from which the distance was measured. In order to normalize the results so that the metric can be used to meaningfully compare results between words with differing numbers of letters, the weighted sum is divided by the sum of the weighting factors used in calculating the sum. In one aspect, a default weighting function multiplies each distance by the following factors: 5 for type PEN_DOWN or ANGLE_THRESHOLD; 4 for type PEN_UP; 7 for type DOUBLE_LETTER ; 6 for type DOUBLE_LETTER2; 3 for type PAUSE; 1 for type PAUSE_ANGLE; and 1 for distances measured from a point on the input pattern other than an inflection point. The different weighting factors used for the various inflection point types reflect the differing expected levels of confidence that an inflection point of the given type has been accurately located and that in fact the detected inflection point reflects the fact that a letter of the word being input should be located nearby.

[0056] In another aspect of the system, a letter of a word being scored may also be matched to a point lying along a path segment between two detected inflection points. Separately or in combination, the aspects of identifying the point of matching according to the location of the second difference local maximum, and of adjusting the weighting applied to a match according to the length ratio and/or the value of the second difference local maximum allow the user to enter input paths in an intuitive and flexible manner. The aspects mentioned create a continuum of gestural variation that enable the user to indicate where letters of the intended word appear on the keyboard without having to carefully enter an input path such that a distinct inflection point is created corresponding to each letter.

[0057] In another related aspect, in calculating the scoring metric for a word, for each successive letter of the word, the distance of the letter-key from the path point with which it is matched is adjusted by a factor that is a function of a variety of measurable aspects of the input path, the particular points with which the letter-keys of a word are matched, and the relationships between these matched path points and the locations of the matched letter-keys themselves. This approach provides useful results because it enables each locally calculated matching score to be influenced by aspects of the input path that extend, to a greater or lesser extent, beyond the local neighborhood of the matched locations, in a way that reflects the likelihood that the input path truly corresponds to the word being scored.

[0058] In another aspect, the context in which a word is to be input is used in combination with a language model to modify the frequencies associated with words in the database. In one approach to language modeling, words in the database include an indication of the word's part of speech (noun, verb, adjective, etc.). A model of the syntax of the language of the words in the database (e.g. English) is used to analyze the sequence of words input by the user. At a minimum, this model takes into account only the word previously output. The language model provides information that certain classes of words (or certain specific words) are more likely or less likely to follow the preceding output word. The frequency information stored with each word provides information as to how frequently each word tends to be used irrespective of context. The information provided by the language model then modifies the frequency associated with a word in the context determined by one or more preceding words. In another approach to language modeling, the database includes information about common word collocations – pairs of words wherein one word tends to follow the other. In another embodiment, to limit the size of the database, language modeling information is added only for pairs of words with similar input patterns where the language modeling information helps to determine which of the words with similar input patterns is more likely in the contexts modeled. In another aspect, language modeling information is added only for pairs of words with similar input patterns which both exceed a minimum threshold frequency.

[0059] The frequency information stored with a word in the database provides an indication of how frequently the word tends to be encountered in general usage. In another aspect, the frequency information associated with a word in a particular user's database is modified in accordance with the specific usage of that word by that user.

[0060] In another aspect, the frequency associated with words in the database is adjusted to differ in one or more cases from the actual frequency with which the word is encountered in general usage. This is done by further lowering the frequency associated with lower frequency words that have input patterns that are similar to the input pattern of a high-frequency word. This improves the likelihood that a high-frequency word will, with high probability, be identified as the most likely matching word when the user tries to enter an input pattern corresponding to the high-frequency word. In another aspect, when the potentially conflicting word is a very low frequency word, or is determined to have an input pattern that is particularly close to the input pattern of a high-frequency word, the word may be eliminated from the database. In one embodiment, an input pattern is created by automatically drawing a path that connects in sequence the centers of the letter-keys corresponding to the spelling of a high-frequency word. A smoothing process is then applied to this input path so that it more closely resembles an input path that has been manually drawn with a stylus. This input path is then processed by the system, and lower frequency words in an initial database that receive a numerical score that is within a threshold difference of the numerical score of the high-frequency word itself are re-assigned a lower frequency in a final database. In another embodiment, the input path is then processed by the system such that, in computing the numerical score for other lower frequency words, when the distance measured for a given letter-key of a lower frequency word from the created input path exceeds a given threshold distance, that threshold distance is subtracted from the measured distance before calculating the numerical score. In another embodiment, the threshold distance subtracted from the measured distance is added to the measured distance for the closest letter-key of the high-frequency word in calculating a numerical score for the high-frequency word with which the numerical score of the lower frequency word is to be compared. In yet another embodiment, the threshold distance varies with respect to whether the measured distance is measured in a direction which is essentially perpendicular to a tangent to the input path at the point from which the distance is measured, or whether the distance is measured from one of the endpoints of the input path in a direction which is essentially parallel to the initial or final segment, respectively, of the input path. This approach allows the adjustment of frequencies assigned to potentially competing lower frequency words to account for the wide variety of input paths that may be entered for a given high-frequency word.

[0061] In another aspect, the input pattern analysis component analyzes the input pattern as the pattern is being input, so that the pattern matching component can begin to identify potentially matching candidate words with little or no delay once the stylus is lifted from the screen. The location at which the screen is first contacted is recorded within the system as the first detected inflection point, and is identified as an inflection point of type PEN_DOWN. In the case where the stylus is again lifted from the screen without being moving more than a threshold distance or remaining in contact with the screen for more than a threshold time period, this first inflection point is recorded as a solitary inflection point of type TAP. In an alternate aspect, as contact location data is received from the touch-screen controller, it is immediately processed through a smoothing algorithm to remove any jitter introduced by the touch-screen digitizer. The path data is then used to calculate the first and second difference of the data stream in both the x- and y-coordinate. The stream of second difference data is then passed through a filter to determine when the sum of the absolute magnitudes of the x- and y- second differences exceeds any of the one or more thresholds determined for an inflection point of type ANGLE_THRESHOLD. Each time any such threshold is exceeded, an inflection point of type ANGLE_THRESHOLD is identified at the contact location determined by the data point at which sum of the absolute magnitudes of the second differences attains its maximum value prior to once again falling below the exceeded threshold. When the stylus is finally lifted from the touch-screen, the location at which the screen is last contacted is recorded within the system as the last detected inflection point, and is identified as an inflection point of type PEN_UP.

[0062] After the input pattern analysis component identifies the inflection points associated with an input pattern, the pattern matching component examines the words stored in the system database to determine which words are the most likely matching candidates. While the aspect described herein is a simple and computationally efficient method to identify which words of a database best match an input pattern, it is to be understood that other alternative approaches could achieve this goal, and should not be considered to be outside the scope of the present invention. In another aspect, the words in the database are organized in a fashion to conserve the amount of storage required while also facilitating efficient searching by the pattern matching component.

[0063] Another characteristic of the input pattern that can be easily and reliably identified by the input pattern analysis component is the total length of the path of the input pattern. The length of the input path cannot be reliably associated with the length of a word in terms of the number of letters in the word, since a word could have a small number of widely separated letters (e.g. “ape”) or a larger number of closely spaced letters (e.g. “deceased”). However, the average expected length for each word is easily calculated for any given keyboard layout and on-screen configuration. In an alternate aspect, the expected path length for a word is calculated as the sum of the distances between the centers of the keys associated with the letters of the word in sequence. In the example of the word “ape,” this would be the sum of the distance from the “a” key to the “p” key, plus the distance from the “p” key to the “e” key. Since the expected path length is only an approximation of what to expect from an actual input pattern for a given word, in another aspect, the range of expected path lengths associated with the words of a database are divided up into a relatively small number of ranges, each such range being associated with a class of words whose expected path length falls into that range. The expected input path length class associated with each word can then be stored along with the word without significantly increasing the size of the database, or alternatively, words in the database can be stored in groups according to expected input path length class. In either case, the number of words that need to be examined in detail by the pattern matching component is greatly reduced since the actual length of the input pattern as measured by the input pattern analysis component is used by the pattern matching component to simply pass over words belonging to an expected input path length class that is significantly different. In an alternate aspect, the word selection component calculates a running average of the ratio of the actual measured length of the input pattern to the expected input path length class of the word selected for output, and the pattern matching component uses this ratio to determine which expected input path length classes will be examined for a given measured input pattern path length.

[0064] In another aspect, the distance from the input pattern to any potentially matching letter-key is compared to a maximum matching threshold distance, such that whenever a letter-key is found to lie further from any possible matching point in the input pattern than this threshold distance, the word is eliminated as a possible candidate. In one embodiment, this serves to speed up the evaluation process, since as soon as the pattern matching algorithm identifies a letter of a word that is outside the threshold distance from any possible matching point on the input pattern, the algorithm immediately proceeds to evaluate the next candidate word. In another embodiment, this approach is used to identify the number and type of instances where a candidate word fails to match an input pattern, such that these instances form the basis of assessing a penalty in determining the scoring metric value that reflects how closely the word is determined to match the input path. In this and in other embodiments disclosed elsewhere, each maximum matching threshold distance used is a function of the object from which the distance is measured (for example, different types of inflection points, or points that lie on path segments between detected inflection points), and may be further modified from any default value as a function of other parametric characteristics of the input path (for example, the average pen speed). Thus, as used in this disclosure, the phrase "maximum matching threshold distance" or "a predetermined maximum matching threshold distance" does not in general refer to a single, unique value, but is specific to each particular instance in which it is applied.

[0065] In another aspect, when the system determines that the stylus has been moved with a greater than average speed in the course of entering an input path, then the input path can be even less precise, in proportion to the increase in stylus speed. In one embodiment, this is achieved by increasing the maximum matching threshold distance between a point on the input path and the letter-key of the word with which the input path point is matched. This threshold distance is adjusted for all points of the path in proportion to the average stylus speed along the entire input path, or in another embodiment, it is adjusted separately for each point that is matched with a letter-key of a word based on the average stylus speed in the neighborhood of the matched point. Similarly, a greater than average stylus speed in the course of inputting a word results in a proportionately greater preference toward matching words of higher frequency. In a similar line of reasoning, to the extent that the entered input path is longer and/or has a higher than average number of inflection points detected, then the input path can be proportionately less precise, since the more letters there are in the intended word the less likely the input path will closely match any other word. Thus, in another aspect, the amount by which the letter-to-path threshold distances are increased is a function of the total length of the input path. In another aspect, the amount by which the letter-to-path threshold distances are increased is a function of the number of inflection points identified in the input path. In another aspect, when the scoring metric for the most-closely-matching word falls below a determined threshold, then the letter-to-path maximum allowable threshold distances are increased, and the database of potentially matching words is searched again using these larger letter-to-path threshold distances.

[0066] In another aspect, when the distance from a determined inflection point of the input path to the next corresponding letter-key of a word is greater than the determined maximum matching threshold distance, the word is retained as a word candidate and a scoring penalty is applied to the word that is used in computing the scoring metric for the word (hereinafter, the “skipping penalty” since the inflection point is “skipped” in the matching process for the word). In another aspect, an additive skipping penalty is assessed against the word being scored by adding a determined weighted penalty amount to the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched in calculating the scoring metric for the word. In another aspect, a multiplicative skipping penalty is assessed against a word being scored by increasing a determined multiplicative penalty factor by which the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched is multiplied when calculating the scoring metric for the word. This skipping penalty approach enables the system in a significant number of cases to correctly identify the intended word despite significant errors on the part of the user in entering the input pattern.

[0067] In another aspect, skipping penalties are also assessed when a letter-key of a word is greater than the determined maximum matching threshold distance from any potentially corresponding point of the input path (hereinafter, a “letter-skipping penalty”). This corresponds to a letter-omission spelling error in tracing out the input path for an intended word. In an aspect analogous to that for inflection point skipping penalties, an additive letter-skipping penalty (or, in the case of a letter substitution, an additive letter-substitution penalty) is assessed against the word being scored. In another aspect, a multiplicative letter-skipping penalty (or, in the case of a letter substitution, a multiplicative letter-substitution penalty) is assessed against the word being scored by increasing a determined multiplicative penalty factor by which the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched is multiplied when calculating the scoring metric for the word. In another aspect, a multiplicative letter-transposition penalty is assessed against the word being scored by increasing a determined multiplicative penalty factor by which the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched is multiplied when calculating the scoring metric for the word.

[0068] In another aspect, the method used to determine the distance of a letter-key from certain types on inflection points is adjusted to accommodate the commonly observed tendency of many users to "overshoot" an intended letter in tracing out an input path. In another aspect, the location determined as the final point of the input path is adjusted from the actual final data point received from the touch-screen digitizer to accommodate certain types of erroneous or misleading data commonly received at the point at which the stylus is lifted from the touch-screen.

[0069] In another aspect, when an inflection point is detected that follows by less than a determined threshold distance along the input path from an immediately preceding inflection point, the following inflection point is not recorded and is treated as an "ordinary" path point. This approach accomodates a tendency common among many users where the stylus is occasionally moved in an erratic manner when the direction of movement is significantly changed, or immediately after first making contact with the touch-screen. It also accommodates the user by making it possible to indicate the location of a desired letter along an otherwise relatively straight portion of the input pattern by moving the stylus in a small "zig-zag" motion on or near the location of the intended letter-key. This approach is advantageous since, with a small amount of practice, such a zig-zag motion can be executed more quickly than a pause, and thus can be used to more rapidly enter an input pattern that is intended to correspond to a lower-frequency word for which the input pattern would otherwise closely match the input pattern for a higher frequency word. As in the previously mentioned example, where the words "pot," "pit," "put" and "pout" all correspond to a straight-line path from "p" to "t," and where "pout" is the lowest frequency word of this group, the user can easily enter the word "pout" by contacting the touch-screen near the letter-key for `p' and making a small zig-zag motion near both the `o' and `u' before lifting the stylus near the `t'.

[0070] The problem of identifying the optimal matching between the M letters of a candidate word and the N identified inflection points and N-1 input pattern path segments is a variant of the "shortest path" problem which is well known in the field of dynamic programming. Various algorithms, such as the Floyd-Warshall algorithm, have been designed to solve the problem of finding the shortest path that traverses an edge-weighted graph from a designated start vertex to a designated end vertex. This classic problem is analogous in certain ways to the problem of identifying an optimal matching between the inflection points and intervening path segments of an input pattern and the locations of the keys associated with the letters of a potentially matching candidate word. Such algorithms are relatively complex and time-intensive (on the order of N^3 for a graph with N vertices). However, the current problem is rendered much more tractable due to a number of important additional constraints:

1. The first and final letters of a candidate word must be matched with the PEN_DOWN and PEN_UP inflection points, respectively, and the sequence of the letters in the spelling of the word must be matched in the sequence in which inflection points occur.
2. The number of letters that can potentially match a given inflection point or path segment is restricted to letters that occur within MAX_DISTANCE of the point or segment, greatly restricting the number of possible solutions to be evaluated.
3. Once the inflection points have been matched with letters, each remaining unmatched letter must be matched with the path segment connecting the inflec-

tion points matched with the closest preceding and following matched letters.

[0071] The fact that the nature of the input method generally results in input patterns of N inflection points for words of M letters, where M is generally not greatly larger than N, combined with the above constraints, makes it possible to design a matching algorithm based on certain heuristics that exhibits a minimal amount of recursion in operation and consequently runs effectively even on devices with limited processing power. Foremost among these is the fact that, for the (N-2) inflection points that must be matched with letters (since the matching for the PEN_DOWN and PEN_UP inflection points is already determined to be the first and last letters of the word, respectively), there are at most (M - N) letters that even need to be considered.

[0072] In one aspect, the pattern matching component uses an algorithm based on the following approach to match the letters of a candidate word of M letters with the N inflection points and intervening path segments determined for an input pattern. Figure 3D shows a flowchart for this matching algorithm, which is described in detail in the Description of the Embodiment, and which does not perform any recursive processing to match the letters of a candidate word with the inflection points determined for an input pattern. The first letter of the word is always matched with the first (PEN_DOWN) inflection point, and the last letter is always matched with the last (PEN_UP) inflection point. For each subsequent letter, the system determines whether it can be matched with the current (next unmatched) inflection point. If it cannot, or alternatively, if the following letter is in fact a better match with (i.e. is closer to) the current inflection point, then determine whether the current letter can be matched with the current path segment at a point that lies between the previous (already matched) inflection point (or the point at which the preceding letter was matched if it was also matched with the current path segment) and the next unmatched inflection point. If not, then the current word is not a candidate, but if so, match the current letter with the current path segment at the determined point and advance to the next letter to determine whether it can be matched with the still-unmatched inflection point. If however, the current letter can be matched with the current (next unmatched) inflection point, then determine whether the following letter can be matched with either the following inflection point or path segment, and if so, match the current letter with the current inflection point and advance to the next letter and the next inflection point, but if not, determine whether the current letter can be matched with the current path segment at a point that lies between the previous (already matched) inflection point (or the point at which the preceding letter was matched if it was also matched with the current path segment) and if so, match the current letter with the current segment and advance to the next letter to determine whether it can be matched with the still-unmatched inflection point.

[0073] In another aspect, words in the database also include an indication of the frequency of use associated with the word. The frequency of use associated with a word is then combined with the weighted sum of distances in calculating the scoring metric whose value is used to determine the relative ranking of potentially matching candidate words.

[0074] In another aspect, one or more of the words identified as the most likely candidates by the pattern matching component are offered to the user for selection by a word selection component. In one aspect, a predetermined number of the most likely candidates, for example, four, are displayed in a word choice list from which the user can select the intended word for insertion into the text being composed. In another aspect, if the user makes no explicit selection of a candidate word from the word choice list, when a subsequent input action is taken (for example, the user begins to trace out a next input pattern or an explicit character is selected for output), the default word choice (the word deemed to be the most likely candidate) is automatically accepted for insertion into the text being composed. In another aspect, when no action is taken by the user for a threshold period of time following the display of a word choice list, the default word choice is automatically accepted for insertion into the text being composed.

[0075] In another aspect, when a word is to be output for insertion into the text being composed immediately following a preceding output word, a single space is automatically inserted into the output text before outputting the word. This significantly speeds up prose text entry since the user no longer has to explicitly activate a space key between each output word, and can enter one word after another as a succession of input patterns, with the proper spacing between words being automatically generated.

Since a space is not automatically output following every word, when the user wishes to enter punctuation such as a comma or period, there is no need to delete an automatically output space before entering the punctuation.

[0076] In another aspect, the system records the length of time that the stylus is in contact with the touch-screen while tracing out the input pattern. This enables the system to estimate the average speed with which the stylus was moved by the user in tracing out the input pattern. Since it is reasonable to assume that, with practice or familiarity, the user will become faster at tracing out the input pattern for a word, the system can use apply a higher weighting to the influence of word frequency in calculating scoring metric values for words in the database, since words that are more commonly used will tend to be entered more quickly that words which are less familiar to the user.

[0077] In another aspect, additional time information is recorded with the input pattern, such as time stamping the sequence of recorded contact locations at fixed intervals (e.g. every 10 msec.) so that the speed with which the stylus was moved along the input pattern at each point can be estimated.

[0078] There will be occasions when the user recognizes that he or she has made so many errors in inputting the current word, such as forgetting a letter, realizing that the stylus has been moved to an unintended location, or some other similar error, that the user concludes the system will be unable to recognize the intended word. In another aspect, the input pattern analysis component recognizes a "Cancel" gesture that can be made at any point in tracing out an input pattern, wherein when the Cancel gesture is recognized, the user is informed by means of an auditory and/or visual display, the current input path is discarded, and no word is output or selection list displayed in response.

[0079] In another aspect, a single tap on the keyboard is recognized by the input pattern analysis component as an inflection point of type TAP. The pattern matching component processes an inflection point of type TAP by first determining a default letter associated with the key within whose boundaries the location of the tap contact occurred (the "tap location letter"). As the user proceeds to continue tapping the keyboard, then the sequence of taps generates a word object at the text insertion point comprised of the tap location letters concatenated in the sequence that the corresponding keys are tapped (the "tap location word"). In another aspect, if this word is not deleted from the output text being generated by the user before typing a space or proceeding to trace out an input pattern to input a next word, if the word is not already present in the database it is automatically added to a user word list of words added by this user to the database of words initially present in the system.

[0080] In many languages, a variety of alternate forms of certain letters are commonly used, generally consisting of letters with various diacritic marks. For example, depending on the language, other forms of the letter "e" that are commonly used include the letters "è," "é," "ê," and "ë" (other forms are also possible - this is simply an illustrative example). Most on-screen keyboards require the user to select a special alternate mode to display one or more alternate keyboards from which the desired alternate letter form must be selected. In the present invention, such alternate letter forms are handled in three distinct and advantageous ways. First, each alternate form of a letter that is used in the language (or languages) of the words in the database is associated with the same key with which the base form of the letter is associated ("e" in the current example). Since the information specifying which alternate forms of letters are used in the spelling of words is included in the database, the user is able to enter such words with alternate letter forms just as words without alternate letter forms are entered – that is, simply by tracing out an input pattern that passes through or near the sequence of keys associated with the base forms of all the letters in the word. In a case where two words are present in the database corresponding to exactly the same sequence of keys (i.e. words that are identical except for the occurrence of one or more alternate letter forms), these words will generally both be added to the word choice list, with the word having a higher frequency of use appearing higher in the word choice list.

[0081] The second manner in which alternate letter forms are handled in an advantageous way is in the entry of new words composed of a sequence of tap location letters. In the present invention, the user does not need to select an alternate keyboard in order to explicitly select an alternate letter form to spell out a new word that is not present in the database. In another aspect, alternate letter forms can be selected by contacting the key with which the base form of the letter is associated and maintaining contact with the key (without sliding off of it) for a period of time exceeding a predetermined threshold length of time. Once the threshold time period is exceeded, a “pop-up list” of the alternate letter forms associated with the key is displayed, at which time the user can slide the point of contact to the desired alternate form of the letter in the list, lifting the stylus to select the alternate form as the tap location letter. This alternate letter form is then added as usual to the tap location word, enabling the user to easily create any desired sequence of letters in spelling a new word, including alternate letter forms, without having to change the mode of the keyboard. The third manner in which alternate letter forms are handled in an advantageous way is described in the detailed embodiment, wherein a first and second default letter-form are defined for a letter-key.

[0082] In another aspect of the invention, the keyboard layout is modified to enhance the system's ability to distinguish between input patterns that might otherwise be more difficult to distinguish. Due to its familiarity to the vast majority of users, the standard “QWERTY” keyboard arrangement is by most users. A disadvantage of this layout is the proximity of the vowels “u,” “i,” and “o.” In another aspect, additional “inactive” areas are defined within the layout of the keyboard, effectively increasing the separation between the centers of certain otherwise adjacent keys, such that when the user attempts to quickly position the stylus near one of the keys, there is on average a greater distance from the stylus position to the other key. In another aspect, the pairs of letter-keys between which inactive areas are placed in a virtual keyboard used for a particular target language is determined by analyzing the frequency of word pairs in the database for the target language which are identical with respect to the sequence of letter-keys with which each word of the pair is associated except for pairs of adjacent (“interchangeable”) letter-keys. In another aspect, the keyboard layout is further modified to enhance the system's ability to correctly analyze input patterns by stretching the keyboard in the vertical direction.

[0083] In another aspect, the system allows the user to select a word in the output text for re-editing, for example by highlighting the word to be edited or by positioning the text cursor within the boundaries of the word and activating a designated editing function key. This causes the system to display a set of possible alternative word choices corresponding to the selected word, such that selecting one of the alternate word choices replaces the selected word in the output text.

[0084] Although the description above contains many specifications, these should not be construed as limiting the scope of the invention but as merely providing illustrations of some of the present aspects of this method. For example, the input pattern can be analyzed with other methods, the database of words can be organized in other ways, and the pattern matching component can use other algorithms to identify the most likely candidate words. For example, on a device with sufficient processing power, the sequence of letters forming each word in the database could simply be compared to the input pattern, measuring in sequence the distance of each letter of a word from the nearest point on the input pattern that occurs later than the point from which the preceding distance measurement was taken. The various important insights embodied in the invention enable text to be input using a touch-screen keyboard rapidly and efficiently using a familiar and constant keyboard arrangement, without having to lift the stylus from the touch-screen between entering each letter, and without having to pause or perform any other action than trace out a path that passes through or near each letter in sequence. The embodiments of the present invention enable the method disclosed to detect and appropriately respond to characteristics of the user's input actions that naturally result of the way that users will tend to interact with the input system. For example, users in general will tend to enter input patterns more rapidly for words which they have entered more often or with which they are more familiar. This corresponds to words which tend to be used with higher frequency. The embodiments of the method also not only allow for the fact that input patterns that are entered with greater speed tend to be entered less accurately, but also take advantage of the contra positive, that input patterns that are entered with lesser speed tend to be entered more accurately. Any type of touch-screen may be used by the system, and the input device may be a stylus, a finger, or any tool that works as an input device on the touch-sensitive screen. The touch-sensitive screen may be utilized by any type of computer or hand-held computer capable of performing the required processing. Thus the scope of the method should be determined by the appended claims and their legal equivalents, rather than by the specific aspects described above.

Brief Description of the Drawings

[0085] The embodiments of the present invention are described in detail below with reference to the following drawings:

[0086] Figure 1 is a hardware block diagram showing the typical hardware components of a system which embodies the method of the present invention such as that shown in Figures 2A through 2E;

[0087] Figure 2A is a schematic view of an embodiment of a portable computer with a touch-screen display on which a keyboard system of the present invention is displayed;

[0088] Figure 2B is the same view showing an embodiment of a word selection list displayed after a user has completed tracing out an input pattern and has lifted the stylus from the touch-screen;

[0089] Figure 2C is the same view showing an embodiment of an alternate letter form pop-up selection list displayed after a user has touched the stylus on the “e” key, and maintained contact with the key past a predetermined time threshold;

[0090] Figure 2D is the same view showing the result of activating an embodiment of the “Re-Edit” function that assists the user in correcting a previously output word when the user failed to select the intended word from the word selection list;

[0091] Figure 2E shows the same view, showing a word selection list displayed after a user has completed tracing out an input pattern very similar to that shown in Figure 2B;

[0092] Figures 3A through 3J show an embodiment of a software algorithm to implement the method of the present invention in order to determine the most likely word or words in a database that match an input pattern traced out by a user;

[0093] Figures 4A-C show a portable computer with a touch-screen display that depicts a possible input path for the same intended word ("pout");

[0094] Figure 5 shows a portable computer with a touch-screen display that depicts an input path corresponding to an embodiment of a shift gesture;

[0095] Figure 6 shows a portable computer with a touch-screen display that depicts an input path which has caused the system to display an embodiment of a visual feedback feature; and

[0096] Figures 7A and 7B show a portable computer with a touch-screen display that depicts an input path which has caused the system to display an embodiment of another visual feedback feature and an embodiment of a selection list sorting algorithm.

DETAILED DESCRIPTION OF THE EMBODIMENT

[0097] Figure 1 shows a simplified block diagram of the hardware components of a typical device 100 in which the System and Method for Continuous Stroke Word-Based Text Input is implemented. The device 100 includes a touch-screen 120 provides input to the CPU (processor) 110 notifying it of contact events when the screen is touched, typically mediated by a hardware controller that interprets the raw signals received from the touch-screen and communicates the information to the CPU 110 using a known communication protocol via an available data port. Similarly, the CPU 110 communicates with a hardware controller for a display 130 to draw on the display 130. Optionally a speaker 140 is also coupled to the processor so that any appropriate auditory signals can be passed on to the user as guidance (predominantly for error signals). The processor 110 has access to a memory 150, which may include a combination of temporary and/or permanent storage, and both read-only and writable memory (random access memory or RAM), read-only memory (ROM), writable non-volatile memory such as FLASH memory, hard drives, floppy disks, and so forth. The memory 150 includes program memory 160 that contains all programs and software such as an operating system 161, a Continuous Stroke Word-Based Text Input software 162, and any other application programs 163. The memory 150 also includes data memory 170 that includes the word database(s) 171 required by the Continuous Stroke Word-Based Text Input software 162, storage for maintaining a record of user options and preferences 172, and any other data 173 required by any element of the device 100.

[0098] In one aspect of the invention, the keyboard layout is modified to enhance the system's ability to distinguish between input patterns that might otherwise be more difficult to distinguish. Due to its familiarity to the vast majority of users, the standard “QWERTY” keyboard arrangement is preferred by most users. A disadvantage of this layout is the proximity of the vowels “u,” “i,” and “o.” Since the system is designed to allow the user to be imprecise in tracing out an input pattern, and since these vowels are often interchangeable in otherwise identical words (e.g. “hut,” “hit,” and “hot”), the proximity of these vowels in the standard “QWERTY” arrangement gives rise to a significant proportion of the incidence of failure to offer the user's intended word as the default choice. In another aspect, additional “inactive” areas are defined within the layout of the keyboard, effectively increasing the separation between the centers of the three adjacent vowel keys, and as a result making it easier for the user to quickly position the stylus relatively nearer to the intended letter of the three. While these unlabeled key areas are referred to as “inactive,” this is only in the sense that a tap which occurs precisely within such an inactive keyboard area elicits no response from the system. Otherwise, the system still processes any input path points that lie within such an inactive area in exactly the same manner as input path points that fall anywhere else within the virtual keyboard. By keeping the each of the virtual keys at approximately the same size, the user is still encouraged to aim for the center of the key area associated with any letter. This helps to maximize the benefit of inserting inactive areas between keys that are most likely to be interchangeable in otherwise identical word contexts. Since confusion between the “u” and the “y”, or between the “o” and the “p” is not a significant concern, it is not necessary to also expand insert inactive areas between these keys. Similarly, the adjacent consonant pairs “s – d,” “g – h” and “n – m,” are often interchangeable, or are interchangeable in one or more pairs of words which are both frequently used words (for example, “has” and “had”), and in another aspect, inactive areas are placed between one or more of these virtual key pairs to increase the separation between the centers of these keys. The preceding examples all relate to a virtual keyboard and corresponding database for the English language. In another aspect, the pairs of letter-keys between which inactive areas are placed in a virtual keyboard used for a particular target language is determined by analyzing the frequency of word pairs in the database for the target language which are identical with respect to the sequence of letter-keys with which each word of the pair is associated except for pairs of adjacent (“interchangeable”) letter-keys. In another aspect, such an analysis is used to design a new keyboard layout in which such pairs of frequently interchangeable letter-keys are positioned so as not to be adjacent, such that the determined layout does not require inactive areas. In another aspect, the analysis is performed to identify pairs of frequently interchangeable letter-keys with respect to a plurality of target languages such that a single layout is designed which tends to minimize such potential adjacent letter-key confusion across the entire set of analyzed languages. In another aspect, each of the languages in the set of target languages analyzed is associated with a weight that is applied to the frequency determined for each pair of letter-keys, such that the potential letter-key confusion identified for a target language with a higher weighting has a greater effect on the final layout determined than the potential letter-key confusion identified for a target language with a lower weighting. Figure 2A shows a schematic view representative of a typical handheld portable computer 2100 (often called a “personal digital assistant” or PDA) that incorporates on its touchscreen 2102 display a keyboard 2104 designed and used in accordance with the present invention. The keyboard 2104, when used in accordance with the present invention, generates text which is output to the text display region 2106 at a text insertion location 2108. The term “keyboard” in this application refers to any keyboard that is implemented on a touch-sensitive surface, including both a keyboard presented on a touch-sensitive display as in Figure 2A, and also a keyboard imprinted on a touch-sensitive surface. Keyboard 2104 explicitly shows the 26 letters of the English alphabet on 26 individual keys, arranged in approximately the standard “QWERTY” arrangement found on most keyboards.

[0099] In another aspect, the keyboard layout is further modified to enhance the system's ability to correctly analyze input patterns by stretching the keyboard in the vertical direction. By increasing the distance between adjacent rows of the keyboard, the user is more easily able to quickly trace out an input pattern that correctly positions both line segments and inflection points within the actual row of the keyboard that contains the key of the intended letter. This improves the performance of the pattern matching component since it greatly reduces confusion between candidate words that differ only in letters that are in adjacent rows (e.g. “hot” and “not”). The scoring metric computation by the pattern matching component can then be modified to apply increased weight to the vertical component of the distance between the key of a letter of a candidate word and the location of an inflection point.

[0100] In accordance with one embodiment, between certain pairs of keys, such as the “u” and the “i” keys 2110 as shown on keyboard 2104, an “inactive” area 2112 is placed that significantly increases the separation between these two keys in comparison to the separation between other adjacent keys. Other inactive areas are placed between other pairs of adjacent keys associated with letters that occur in pairs of words that are identical save for the substitution of one of these letters for the other. Thus, similarly, and for the same reason, “inactive” areas 2114 are between the “s” and “d” keys, between the “g” and “h” keys, and between the “n” and “m” keys 2116.

[0101] Text is generated by contacting keyboard 2104 at or near the key associated with the first letter of the word being input, and tracing out a continuous pattern that passes through or near each letter of the word in sequence, and breaking contact with the touch-screen when the last letter of the word is reached. Figure 2B shows the same schematic view of the computer 2100, where the path of a representative input pattern 2200 is shown superimposed on the displayed keyboard 2104. In one embodiment, the user may select as an option whether the path of the input pattern is in fact literally drawn on the display, and erased when a word is selected from a selection list 2208 displayed in the display region 2106, or when the selection list 2208 is not displayed because the difference in the scoring metric calculated for the most-closely matching and second-most-closely matching words falls below a determined threshold, or when the selection list 2208 is canceled. In the example shown in Figure 2B, this option is turned on for illustrative purposes. In this example, the user has attempted to enter the word "text," and the system has successfully matched the word "text" as the most likely candidate word so that it is displayed in a default word choice location 2210 in selection list 2208. The path of an input pattern 2200, as entered by a user using a touch device, such as a stylus pen, starts at an initial contact point 2212, which location is received by the processor and recorded by an input pattern analysis component that is being executed by the processor as the PEN_DOWN inflection point for the input pattern. In this example, the user moves the stylus so that the path then moves along path segment 2226 first to the region of the letter-key associated with the letter "e," then turns sharply to move along path segment 2228 in a direction that is only approximately toward the region of the letter-key associated with the letter "x," creating an ANGLE_THRESHOLD inflection point that is recognized by the input pattern analysis component at location 2214. Then, in the vicinity of (though not on) the key associated with the letter "x," the path turns sharply back up toward the letter-key associated with the letter “t,” creating a second ANGLE_THRESHOLD inflection point that is recognized by the input pattern analysis component at location 2216 within the bounds of the letter-key associated with the letter “z.”. Finally, after traveling along path segment 2230, the stylus is lifted from the touch-screen at location 2218, which is recorded by the input pattern analysis component as a PEN_UP inflection point for the input pattern. In another embodiment, the selection list 2208 also shows three additional candidate words having the next three highest scoring metric values, which in an example shown in Figure 2B are the words “great,” “test,” and “treat.” In the embodiment, when a letter of a word lies more or less along a straight path between the preceding and following letters (as in the case of the letter “r” in “great”), there is no need to do anything other than continue the path of the input pattern through or near the intended letter. Thus, in accordance with another embodiment, in the example of Figure 2B, although only four inflection points were identified by the input pattern analysis component, the processor identifies the words “great” and “treat” (which have more than four letters) as potentially matching candidates since the four inflection points identified are within the determined maximum matching threshold distance of the letters “g” (or “t” in the case of “treat”), “e,” “a” and “t.”.

[0102] In another embodiment, the selection list 2208 also includes a “(more)” function 2220, the selection of which causes the processor to identify and display the next four additional candidate words having the highest scoring metric values of the remaining words in the database. In the example of Figure 2B, although not shown, the next four such candidates are “gear,” “year,” “rest” and “heart”, and would be displayed in selection list 2208 in response to a selection of the “(more)” function 2220. If for any reason the user chooses not to select any word in the displayed selection list 2208, the selection list display can be closed by selecting a “CANCEL” function 2222. In another embodiment, activating a back-space or delete key also cancels a displayed selection list 2208 without selecting any of the displayed words for output.

[0103] In yet another embodiment, when the user enters input patterns with sufficient precision and finds that the default is virtually always the intended word, the user can choose to turn off the selection list display such that only the default word is displayed at the insertion point. If the user wishes to select a word choice other than the already-output default word, a Re-Edit function key 2224 that is presented in the keyboard 2104 is activated before proceeding in order to display a selection list with alternate word choices. In another embodiment, instead of turning off the selection list display entirely, the user can choose a different value for threshold difference between the scoring metric for the most-closely-matching and the second-most-closely-matching words that must be exceeded to suppress the display of the selection list, so that the selection list is displayed less frequently. Alternatively, in another embodiment, the user can simply choose to reduce the number of word choices displayed in the selection list 2208.

[0104] In accordance with another embodiment, although not normally displayed explicitly on the keys of keyboard 2104, various alternate letter forms, such as letters with diacritic markings, are associated with each key that is associated with and displays the base form of the letter with alternate forms. In accordance with another embodiment, Figure 2C shows a “pop-up” menu 2300 of alternate letter forms of the letter “e” that is displayed after a user has touched the stylus on the “e” key, and maintained contact with the key past a predetermined time threshold. In the example shown in Figure 2C, the user has slid the stylus' point of contact 2302 up to the list row 2304 containing the alternate letter form “é,” which is correspondingly highlighted, so that when the user lifts the stylus from the screen, the letter “é” will be explicitly added to the word currently being spelled through conventional “tapping.” This embodiment enables a user to explicitly enter alternate forms of letters to spell out words that are not yet present in the system's database, without having to switch to an alternate keyboard layout display. In the example shown in Figure 2C, the user is in the process of spelling out the word “Café,” and has already “tapped” the Shift key, followed by the “c,” “a” and “f” keys, creating the TAP location word object “Caf” which appears in a word selection list 2306 at the text insertion point as a default (and only) word object in the list. In another embodiment, the each letter associated with each of a sequence of tapped letter-keys is output directly at the text insertion point such that the TAP location word object (in the current example, “Caf”) appears directly in the text being output, so that no selection list is displayed. In this alternate embodiment in which no selection list is displayed, the TAP location word is implicitly selected and no explicit selection action need be performed by the user.

[0105] When the user lifts the stylus from the screen at position 2302, the letter “é” will be appended to the TAP location word to form the word “Café,” which, in accordance with another embodiment, can be explicitly selected by tapping the selection list 2306 at row 2308, or implicitly selected by proceeding to enter a continuous stroke input pattern for a next word. Alternatively, the user can cancel the current selection list by selecting the row 2310 associated with the CANCEL function. In the alternate embodiment in which no selection list is displayed, the user can cancel the TAP location word by deleting it from the output text prior to typing a space (or other character which is not allowed as a character within a database word) or entering an input path for a word to follow the TAP location word in the output text. In another preferred embodiment, when the stylus taps on a letter-key and remains in contact with the key past the predetermined time threshold, a menu of the alternate letter forms associated with the letter-key is displayed so that the user can lift the stylus from the letter-key and tap the displayed menu on the desired alternate letter form to select it for output at the text insertion point.

[0106] When the default word choice presented by the system corresponds to the user's intended word, no additional action is required from the user, since even when a selection list is displayed, proceeding to trace the input pattern for the next desired word causes the default word choice to be automatically output for insertion into the text being composed. When the system correctly identifies the user's intended word with a high degree of accuracy, there may be a tendency for a user to pay less attention to the word choice list in order to move quickly from one word to the next, speeding up the entry of text. As a result, there may occasionally be instances where the default choice does not correspond to the user's intended word, so that an unintended word is entered into the output text which must be subsequently edited by the user to change it to the intended word. In another aspect, the system allows the user to select a word in the output text for re-editing, for example by highlighting the word to be edited or by positioning the text cursor within the boundaries of the word and then activating a designated “Re-Edit” function key. In one embodiment, when a word is selected for re-editing, the system creates a simulated input pattern by creating a path that connects in sequence the centers of the keys associated with the sequence of letters that comprises the word. In order to avoid creating “artificial” inflection points where they were unlikely to exist in the original input pattern, a smoothing process is first applied to the created input pattern to avoid the creation of ANGLE_THRESHOLD inflection points at keys where in fact the path of the input pattern changes direction only slightly. The smoothed input pattern is then processed by the system in the same manner as an input pattern traced out by the user, except that the word selected for re-editing is filtered out and does not appear in the displayed list. In another aspect, a longer word choice list is created to increase the probability that the user's originally intended word appears somewhere in the word choice list. Since the word being re-edited was close enough to the original input pattern to be selected as the default word choice, there is a reasonable likelihood that the originally intended word will be close enough to the created input pattern that it will appear in the word choice list created by the pattern matching component in processing the created input pattern. Selecting a word from the word choice list automatically replaces the word highlighted for re-editing with the selected word.

[0107] Figure 2D shows how the Re-Edit function can be activated by the Re-Edit function key 2224 to correct a previously output word when the user has unknowingly accepted the default word for output to the text area 2106 in an instance where the default word did not correspond to the intended word. The unintended output word is selected, either by double-tapping the word to highlight it or by using any of a number of well-known techniques. Once the target word ("great" 2400 in Figure 2D) is selected, the user activates the Re-Edit function key 2224 by tapping it. In another embodiment, when the Re-Edit function key 2224 is tapped and no word is currently highlighted, the system identifies the word containing or adjacent to the current text cursor location and automatically selects it as the target of the Re-Edit function. In another aspect, the system maintains a record of a predetermined number of the most recently output words, wherein the record includes for each output word the ordered list of words identified as the most likely matching words with respect to the input path that resulted in the originally output word. When the Re-Edit function key 2224 is tapped, the system then displays a word choice list containing the list of words originally identified as the most likely matching words in the order determined by the scoring metric values calculated with respect to the original input path entered by the user. In another preferred embodiment, the originally output word is omitted from the displayed word choice list since the Re-Edit function is only activated in order to replace it. Figure 2D shows the resulting selection list 2402. In this example, the originally intended word "heat" appears as the first word 2404 in the selection list 2402 because it was determined to be the second-most-closely matching word with respect to the original input path (following the word "great" which was originally output as the default word). Selecting the word 2404 in the selection list 2402 automatically replaces the highlighted target word "great" with the originally intended word "heat" in the output text area 2106.

[0108] To illustrate another aspect of the invention, Figure 2E shows another input path 2500 for comparison with the path 2200 shown in Figure 2B. In this example, the user has attempted to enter the word "great," and the system has successfully matched the word "great" as the most likely candidate word so that it is displayed in a default word choice location 2510 in selection list 2508. The path of the input pattern shown in Figure 2e starts at an initial contact point 2512, which location is virtually identical to the initial contact point 2212 of Figure 2B. As a natural result of the user's intention to enter the two letters “r” and “e” following the initial letter “g” (corresponding to the intended word great), the initial path segment 2526 passes through both of the corresponding letter-keys before continuing to curve downward toward the general region of the letter-key “a.” A significant benefit of the present invention is the fact that the user does not need to strive to make the input path conform to any particular pattern - the user is free to simply trace out any pattern that represents a reasonable attempt to traverse the keyboard in sequence near the letter-keys of the intended word. Thus, as shown in the representative input path of Figure 2E, while the user is free to intentionally create an ANGLE_THRESHOLD inflection point at any desired location (to increase the likelihood that the intended word will be correctly identified by the system as the most likely matching candidate), there is no requirement to artificially create a ANGLE_THRESHOLD inflection point near either of two letter-keys for “r” or “e,” and consequently no inflection point has been created in the example input path shown. The first ANGLE_THRESHOLD inflection point detected by the analysis of the input path shown occurs at location 2516, where having reached the region of the letter-key associated with the letter “a,” the user begins to move the stylus toward the region of the letter-key associated with the letter “t,” the final letter of the intended word. In consequence, while the initial path segment 2526 does not have a detected inflection point in the region of the letter-key associated with the letter “e” (as is the case with the input path shown in Figure 2B), it does exhibit a distinct curvature so that the length of the input path segment 2526 from location 2512 to the inflection point at location 2516 is significantly greater than the straight-line distance between those locations. Note that, except for the path segment curvature just noted, and the resulting absence of any ANGLE_THRESHOLD inflection point on input path segment 2526, which is in contrast to the corresponding but relatively straight path segments 2226 and 2228 of Figure 2B with the intervening ANGLE_THRESHOLD inflection point 2214, input path 2200 of Figure 2B and input path 2500 of Figure 2E are extremely similar in that: 1) the initial and final contact locations (PEN_DOWN locations 2212 and 2512, and PEN_UP locations 2218 and 2518, respectively) are virtually identical in both cases; 2) the points of closest approach to each of the significant letter-keys (“t”, “g”, “r”, “e”, “a”, “x” and again “t”) are virtually identical in both cases; and 3) the locations of the ANGLE_THRESHOLD inflection points 2216 and 2516 and in consequence their distances from the letters “a” and “x” are also virtually identical in both cases. The combination of the various aspects described below produce a surprising result: that the characteristics of various input paths that are reliably detected by the analysis methods of the present invention in a computationally very efficient manner arise as a result of a user's natural and intuitive approach to tracing out input paths for words.

[0109] In another aspect, when a word is generated for output immediately following a previously output word, the system automatically outputs a space prior to outputting the generated word, eliminating the need to perform any additional action for the great majority of space characters output. Similarly, when a word is generated for output immediately following one of a set of punctuation characters and other characters that are normally followed by a space, the system also automatically outputs a space prior to outputting the generated word. For example, when a string of one or more digits is typed, a space is usually entered before any following word. However, in another aspect, certain words are designated such that no space is automatically generated prior to outputting one of the designated words after one or more of the designated characters. For example, following a string of digits with any one of the four words “st”, “nd”, “rd” or “th” (as in the strings “1st”, “22nd”, “103rd” or “11th”) suppresses the automatic generation of an intervening space such that the intended text output is generated.

[0110] Note that, in the case of a period, with the prevalence of the use of the period in entering URLs and e-mail addresses (in which no space generally appears following a period), automatically spacing after each period can be counter-productive, since there is a significant effort involved in editing the text to remove an unwanted space that appears in the text output at a position no longer adjacent to the text-output cursor location. In another aspect, the system does not automatically generate a space following a period. The database of words is augmented to include designated “punctuation words” comprised entirely of punctuation characters and/or spaces, and wherein one or more of these words are marked in the database such that, when they are generated as output by the system, no space is automatically generated preceding or following them. For example, to facilitate the entry of an end-of-sentence period, the database includes a two-character textual object comprised of a period followed by a space. Following the output of the final word of a sentence, the user simply contacts the keyboard in the vicinity of the key associated with a period and moves the point of contact to the vicinity of the key associated with a space before lifting the stylus. This generates a period and space immediately following the final word of the sentence, creating a well-formed sentence according to the user's intention. To facilitate the entry of an end-of-sentence period followed by two spaces, the database includes a three-character textual object comprised of a period followed by two spaces, which the user enters by contacting the keyboard in the vicinity of the key associated with a period and moving the point of contact to the vicinity of the key associated with a space, and performing a “double-letter gesture” (as described below) before lifting the stylus. In another embodiment, when the most closely matching word is a “punctuation word” (as described previously), then the punctuation word is automatically accepted as the textual interpretation of the input pattern and is generated as the output of the system even when the difference between it and the second-most-closely matching word does not exceed a determined threshold.

[0111] Figures 3A through 3J show a process flow chart of an embodiment of software 162 to implement the Method for Continuous Stroke Word-Based Text Input that generates and manages a word selection list in response to the user contacting the screen and entering a continuous stroke input pattern or a tap contact. Figure 3A shows a flow chart of an embodiment of the Main Processing Routine 3100 of the Continuous Stroke Word-Based Text Input software 162. At block 3105, when the process is first started, various system variables are initialized. At block 3110, the process waits to be notified that a contact has occurred within the area of the keyboard 2104. When the operating system detects a contact in the keyboard area 2104, control returns to the Main Processing Routine 3100 at block 3115, where an Input Pattern Analysis Routine 3200 of Figures 3B-1 and 3B-2 is invoked to analyze and classify the nature of the user's contact action.

Once the contact action has been analyzed, a Pattern Matching Routine 3300 of Figure 3C is invoked at block 3120 to determine what word candidates will be presented, what text is generated, or what function is invoked in response to the analyzed contact action. At block 3125, a Display Selection List Routine 31000 of Figure 3J is called to generate a Word Choice list display to allow the user to select the intended word, if needed. Following the return from the Display Selection List Routine 31000, control returns to block 3110 where the process waits to be notified of the next contact action that occurs within the area of the keyboard 2104.

[0112] Figure 3B shows a flow chart of an embodiment of the Input Pattern Analysis Routine 3200. At block 3205, required variables are initialized, and arrays and corresponding indices used to store an input pattern data are cleared, along with an Inflection Point Table (IPT) in which the determined information about each identified inflection point is stored, such as its type, location, and time of occurrence. In one embodiment, to reduce the number of calculations performed by the Pattern Matching Routine 3300, each inflection point entry in the IPT also includes an array IP_Distance[] that is filled in with the distance from the inflection point to each key of the keyboard that is associated with a letter. In another embodiment, when the distance from an inflection point to a key of the key-board exceeds a predetermined maximum threshold, the corresponding entry in the IP_Distance[] array is set to a unique MAX_DISTANCE flag value. In another embodiment to further reduce the number of calculations performed by the Pattern Matching Routine 3300, each valid distance entry in the IP_Distance[] array for an inflection point is stored in the array as the distance multiplied by a weight-ing factor for the type of inflection point. Similarly, each inflection point entry in the IPT (except the first PEN_DOWN) includes an array Path_Distance[] that is filled in with the distance from each key of the keyboard that is associated with a letter to the closest point on the preced-ing input path segment (between the preceding inflection point and the current inflection point), here again multi-plied by the weighting factor PATH_WEIGHT determined for a distance measured from a path segment (or to the MAX_DISTANCE flag value if this distance is greater than a corresponding maximum matching threshold distance for path segments). In another embodiment, the weighting (PATH_WEIGHT) given to the measured distance of a letter-key from a path segment point to which it is matched is a function of how much the path segment varies from a straight line between the detected inflection points that bound the segment. In one embodiment, this variance is measured as a function of the ratio of the length of the input path segment to the straight-line distance between its endpoints (the "LENGTH_RATIO"). In another aspect, the length ratio is used to adjust the measured distance of a letter-key from the path segment point to which it is matched (prior to applying any weighting factor). In another aspect, when a letter is matched to a point on a path segment, a determined neighborhood of the letter along the path is examined to identify a point where the value of the second difference of the path coordinates reaches a local maximum, and this point is used as the input path point to which the letter is matched. In another aspect, the value of this second difference is used in calculating the weighting applied to the letter matching in the scoring metric calculation. In another aspect, the value of this second difference is also used to adjust the measured distance of a letter-key from the path segment point to which it is matched (prior to applying any weighting factor).

[0113] Separately or in combination, the aspects of identifying the point of matching according to the location of the second difference local maximum, and of adjusting the weighting applied to a match according to the length ratio and/or the value of the second difference local maximum allow the user to enter input paths in an intuitive and flexible manner. The aspects mentioned create a continuum of gestural variation that enable the user to indicate which letter-keys on the keyboard correspond to letters of the intended word without having to carefully enter an input path such that a distinct inflection point is created corresponding to each letter. This allows the user to, when needed, lightly “hop on” or “skip across” intended letters. Figure 4A shows a possible input path 4100 for the word “pout” where the input path has been entered without any attempt to distinguish among the four words “pot,” “pit,” “put” and “pout” which all correspond to a straight-line path from “p” to “t.” The result is in accordance with the relative frequencies of the four words (“pout” is the lowest frequency word and “put” is the highest frequency word of this group), and the particular path which happens to pass closer to the center of the letter-key associated with “i” than to the center of the letter-key associated with “o” (so that “pit” appears in the selection list 4102 before “pot”). The input path 4200 shown in Figure 4B, which is correctly recognized (by an implementation of the present invention running in real-time on a standard Hewlett-Packard iPAQ 2215 PocketPC device) as corresponding to the word “pout” which appears as the most likely (default) word in selection list 4202, is more “flowing,” and exhibits less dramatic changes in path direction. Since each letter of the intended word “pout” lies in the same direction (i.e. a straight-line path from “p” to “t” passes directly through each letter in the proper sequence), the less an input path has to diverge from a straight line, the easier the path will generally be for the user to enter, and consequently, faster. This is in contrast to Figure 4C, which shows an input path 4300 for the same word where a distinct inflection point is created at both the “o” and the “u.” While this type of input path is also correctly recognized by the present invention as most likely corresponding to the intended word “pout” (as seen in selection list 4302), entering the input path as shown in Figure 4C requires four additional sharply angular changes in direction (two each within the regions of the letter-keys associated with “o” and “u,” respectively). For an average user, such an input path will require more effort, and in general more time, to enter than one such as that shown in Figure 4B. The first time a user tries to enter the word "pout," it is likely that the input path entered will be more or less a straight-line path such as that shown in Figure 4A, such that "pout" will have to be selected from a lower position on the word choice list, following the other higher frequency words in the group. This observation can enable the user to enter the "hopping" path of Figure 4A for the word "pout" on subsequent occasions when the word is to be entered. Such experiences tend to prompt the user to enter similar "hopping" paths when they realize that they are entering a word that is both lower-frequency, and short in length.

[0114] Returning to Figure 3B-1, next, at block 3215, the process waits to determine whether the point of stylus contact exits from the region associated with the key in which the first contact location occurred, or whether the stylus is lifted and contact with the screen is terminated prior to exiting from the key (or whether the stylus is kept in contact with the screen within the bounds of the key region for longer than a determined Tap-Hold time threshold), in which case execution proceeds to block 3220 where a single inflection point of type TAP is entered in the IPT. Also, in another embodiment, when the process detects at block 3222 that the point of contact did not exit from the key in which contact first occurred, and the contacted key is associated with one or more alternate letter forms, and a predetermined threshold of time has elapsed since the contact was initiated, then at block 3225 a single inflection point of type TAP-HOLD is entered in the IPT, and control returns from the Input Pattern Analysis Routine without waiting for the contact to be lifted from the touch-screen. When the Pattern Matching Routine receives and processes a TAP-HOLD inflection point, a “pop-up list” of the alternate letter forms associated with the key is displayed, and the process waits until the user slides the point of contact to a desired alternate form of the letter in the list, lifting the stylus to select the alternate form as the tap location letter. A flag is then set so that this alternate letter form is then added to the tap location word (as shown in Figure 3F), and if TAP_Word_Len is currently set to 0, the selected alternate letter form also becomes the default one-letter word in the word choice list. This enables the user to easily create any desired sequence of letters in spelling a new word, including alternate letter forms, without having to change the mode of the keyboard.

[0115] If at block 3215 the contact point exits from the initial key contacted (or a DOUBLE_LETTER gesture is detected), then, moving to Figure 3B-2, at block 3230 a Word Output Routine 3900 (shown in Figure 3I) is invoked to output the default word choice if there is a pending selection list currently displayed from a preceding input pattern. Then at block 3233, in one embodiment, the sequence of all raw unprocessed data points collected up to that time from the touch-screen are appended to the Input Pattern data buffer, and in another embodiment at block 3235, first and second order differences are calculated and stored, and in another embodiment, the sum of the absolute magnitudes of the x- and y- second order differences is calculated and appended to a separate Input Pattern data buffer. Then at block 3240, the process determines if a DOUBLE_LETTER gesture was detected in the sequence of data just processed. In one embodiment, this determination is made based on whether a pattern of path points is detected where either or both of the first-order x- and y-differences change in sign, and where each sign change location occurs within a threshold distance of a keyboard location that is approximately at the center of the set of detected path points, and wherein the sequential pattern of detected sign changes corresponds to a continuous clockwise or counter-clockwise loop. If so, at block 3245, the approximate center of the gesture is determined and added to the IPT as a DOUBLE_LETTER inflection point at block 3250 as described above. Also at block 3250, each time an inflection point is added to the IPT, the weighting factor for the type of the inflection point is also recorded in the IPT.

[0116] Then at block 3255, the process calculates the LENGTH_RATIO for the preceding input path segment (as described previously), and uses this value in calculating the PATH_WEIGHT value to be used in weighting distances to letter-keys that are measured from points along the preceding path segment. At block 3260, the Path_Distance[] table for the segment is filled in with measured distances to which the calculated PATH_WEIGHT has been applied.

[0117] At block 3265, the process determines whether all input path data has been processed up through the location where the stylus was lifted from the touch-screen, and if so, at block 3295 the LENGTH_RATIO is calculated for the final input path segment and stored in the IPT, and at block 3298 the final PEN_UP inflection point is added to the IPT in the manner previously described.

[0118] If no DOUBLE_LETTER gesture was detected in the sequence of data at block 3240, then at block 3270 the process determines whether at some point in the analyzed sequence of data, the sum of the absolute magnitudes of the x- and y- second differences exceeds a predetermined threshold. If so, then at block 3275, the process determines the point where the sum of the absolute magnitudes of the second differences attains its maximum value prior to falling back below the minimum threshold, and this point is then added to the IPT as an ANGLE_THRESHOLD inflection point in the manner previously described.

[0119] There will be occasions when the user recognizes that he or she has made so many errors in inputting the current word, such as forgetting a letter, realizing that the stylus has been moved to an unintended location, or some other similar error, that the user concludes the system will be unable to recognize the intended word. Since the system will always attempt to find the best match and identify at least one default word choice, the user would then have to lift the stylus from the touch-screen, move to a key that cancels the current word choice list, and activate that key. In another aspect, the input pattern analysis component recognizes a “Cancel” gesture that can be made at any point in tracing out an input pattern. In one aspect, the gesture includes simply moving the stylus rapidly back and forth a threshold number of times, where the stylus is moved faster than a threshold speed. In one aspect, the default number of movements is three (for example, right-left-right), and the minimum threshold speed of movement is automatically set in reference to the fastest stylus movement speed measured during input of a word that was accepted by the user (for example, 5% faster than this fastest speed). Since the “Cancel” gesture does not need to be controlled or executed with any precision, it is easy to perform at a high speed). The Cancel gesture described in the present invention is simple and intuitive because it conveys the feeling of simply “scribbling out” the previously drawn input pattern. In another aspect, the system provides visual and/or auditory feedback (for example, a distinctive beep) as soon as the input pattern analysis component recognizes the Cancel gesture. The user can then simply lift the pen from the touch-screen and start over again entering the intended word. This feature is also useful when the user simply decides mid-way through entering a word that a different word is more appropriate.

[0120] In another embodiment, since a Cancel gesture can in one sense be defined as three or more successive ANGLE_THRESHOLD inflection points entered at an accelerated speed, following the detection of each ANGLE_THRESHOLD inflection point, at block 3280 the process checks whether a Cancel gesture has been entered. If so, at block 3285, a CANCEL signal is generated to inform the user that the Cancel gesture has been recognized, the process waits for the stylus to be lifted from the touchscreen, and then the IPT and Input Pattern data buffer are cleared before returning to the Main Routine 3100. If no Cancel gesture is detected at block 3280, the process proceeds to finish adding the ANGLE_THRESHOLD inflection point at block 3250, and proceeds as before for a DOUBLE_LETTER inflection point.

[0121] In another embodiment, each time an inflection point is detected, a test is performed prior to recording the newly detected inflection point in the IPT at blocks 3245 through 3260 (which results in the newly detected inflection point being subsequently used by the Pattern Matching Routine). When the newly detected inflection point follows by less than a determined threshold distance along the input path from an immediately preceding inflection point, the newly detected inflection point is discarded (not recorded in the IPT) and the Input Pattern Analysis Routine continues to process the remaining data points. In one aspect, the threshold distance between detected inflection points is determined with respect to the types of the following and preceding inflection points. In another aspect, a single inflection point is determined at a location midway between the following and preceding inflection points which replaces both inflection points in subsequent processing of the input path.

[0122] In another aspect, in order to reduce the influence of instances where the letter-key of a word being scored happens to be located very close to an inflection point or segment of an input path with which it is being matched, the distance calculated for a letter-key from an inflection point is compared to a determined minimum threshold distance. Thus, in another embodiment, at block 3250 (and at block 3298), when the calculated distance from a letter-key to the inflection point location falls below the determined minimum threshold, the calculated distance is replaced by the minimum threshold distance in the table, so that the minimum distance is used by the Pattern Matching Routine in determining a numerical scoring for a word. In another aspect, the calculated distance is replaced by the average of the actual distance and the minimum threshold distance in determining a numerical scoring for the word. This reduces the likelihood that an unintended word will be placed too highly in the relative ranking of words to one or more letter-keys that happen to be located very near to the input path. In another aspect, the minimum threshold distance used is determined with respect to the type of inflection point with which the letter-key is matched. In another aspect, at block 3260 (and at block 3298), a second distinct minimum threshold distance is determined and used in a corresponding fashion to adjust the calculated distance of a letter-key from a point on a path segment of an input path (as opposed to an inflection point of the input path).

[0123] In another aspect, the method used to determine the distance of a letter-key from certain types on inflection points is adjusted to accommodate the commonly observed tendency of many users to “overshoot” an intended letter in tracing out an input path. This is particularly common in the case of the final letter of a word, where the user may be more confident of the direction in which the path should be traced, and increases the speed with which the stylus is moved. Another factor is that the stylus needs to be pressed down enough to remain securely in contact with the touch-screen throughout the tracing of the input path, and there may be a slight delay in releasing this downward pressure such that the stylus remains in contact with the screen significantly beyond the point at which the final letter-key of the intended word has been traversed. The same phenomenon may occur at an ANGLE_THRESHOLD inflection point, where the path may travel well beyond the point at which an intermediate letter-key of the intended word is traversed before changing direction to proceed toward the next letter-key of the word. In one aspect, since the degree of "overshoot" tends to be more pronounced in the case of the final letter-key of a word, the amount of adjustment applied to the calculated distance is greater in that case. In another aspect, when the distance of a letter-key from an inflection point is greater than the distance from the point closest to the letter-key on the path segment immediately preceding the inflection point, then the distance to be used in the scoring procedure is determined as a weighted average of the two distances. In another aspect, the distance to be used in the scoring procedure is adjusted by decrementing the distance of the letter-key from the inflection point by a determined proportion. In another aspect, an adjustment to the distance determined for the letter-key is made only when the distance of the letter-key from the closest point on the path segment falls below a determined threshold. In another aspect, an adjustment to the distance determined for the letter-key is made only when the distance of the letter-key from the inflection point falls below a determined threshold. In another aspect, an adjustment to the distance determined for the letter-key is made only when the slope of the line from the input path point matched with the preceding letter-key to the path segment point matched with the letter-key differs by less than a determined threshold amount from the slope from the path segment point matched with the letter-key to the inflection point. This ensures that the adjusted distance is in fact an “overshoot” in that it occurs in a direction that is more or less a straight line extension of the path from the preceding matched key.

[0124] In another aspect, the location determined as the final point of the input path (which represents the point at which the stylus is lifted from the touch-screen) is adjusted from the actual final data point received from the touch-screen digitizer. Certain touch-screen digitizers, particularly those commonly found in “TabletPC” devices, make use of “active pen” touch-screens, where the digitizer can sense the presence of the stylus tip even when it is hovering near the screen and not in actual contact with it. In such cases, the data reported when the stylus is removed from contact is unreliable, often exhibiting an erratic trace in a direction unrelated to the final intentional movement of the user. A similar result may be obtained due to the fact that the user's ability to intentionally control the movement of the stylus on the surface of the touch-screen may be reduced when the user lifts the stylus from the screen, at which time the stylus will often move erratically before actually breaking contact with the screen. In another aspect, when the input path changes direction by more than a determined threshold amount at a point of the input path that is less than a determined threshold distance along the remaining path length to the final data point of the input path, then the path point data following the point at which the change in direction is detected is ignored (discarded) and the direction-change location is regarded as the point at which the stylus was lifted.

[0125] Following return from the call to the Input Pattern Analysis Routine 3200, at block 3120 in the Main Processing Routine 3100, the Pattern Matching Routine 3300 is called to process the results of the analysis of the input pattern. As shown in Figure 3C-1, at block 3305 the word candidate table is cleared by setting Num_Candidates to zero, and Max_Metric_Value is initialized to a flag value MAX_FLAG, indicating that the word candidate table is still empty. Then at block 3307, the process checks whether a Cancel gesture was entered, and if so, returns at block 3320 with the word candidate table still empty. If no Cancel gesture was entered, then at block 3310 the process checks whether the inflection point table contains a single TAP or TAP-HOLD inflection point, and if so, at block 3315 the Process TAP Inflection Point Routine 3600 is called to handle the detected TAP input pattern.

[0126] Otherwise, in another embodiment, at block 3323, an adjustment is made to the maximum matching distance thresholds used to determine which letter-keys are greater than the maximum allowed distance from the detected inflection points (and the input path segments between them). With the computational speeds available with modern processors, much if not all of the analysis of the input pattern can be completed during the time that the user is actually entering the input pattern. This analysis that is performed before the pattern has been completed must be carried out with these thresholds set to their default values. Once the input pattern is complete, the results of the initial analysis can be adjusted based on characteristics of the completed input path that are determined through the initial analysis. For example, if the analysis determines that the maximum matching distance thresholds used were too small, letter-key to inflection point distances that were initially flagged as exceeding the maximum matching threshold distance are re-examined with respect to revised larger maximum matching distance thresholds, and reset in the IPT according to the comparison with the revised thresholds. Similarly, if the analysis determines that the maximum matching distance thresholds used were too large, then letter-key to inflection point distances that exceed to revised smaller thresholds are reset to being flagged as exceeding the maximum matching threshold distance.

[0127] In one embodiment, the average stylus speed along the entire current input path is determined and compared to the average stylus speed calculated for a determined number of the previously entered input paths. The result of the comparison (for example, the ratio of the two average speeds) is then used to adjust the values of various analysis parameters that apply to the entire input path. In one embodiment, the maximum matching distance thresholds used for one or more types of inflection points are adjusted in accordance with the comparison, wherein a relatively higher stylus speed for the current input path increases the size of the maximum matching distance thresholds relative to their default values. In another embodiment, as detailed below, a relatively higher stylus speed for the current input path increases the effect that the relative frequencies associated with words in the database has on the final scoring metric determined for potentially matching candidate words, such that words associated with higher relative frequencies are accorded a proportionally greater-than-average advantage in the calculated scoring metric. In another aspect, the local speed of the stylus as it passes through a determined neighborhood of a point on the input path is compared to the average stylus speed along the entire current input path, and the result of the comparison is then used to adjust the values of the maximum matching distance thresholds used for one or more types of inflection points that lie within the determined neighborhood. In another aspect, when the analysis detects that the movement of the stylus paused at one or more locations in the course of entering an input path, the duration of the detected pauses is omitted in calculating the average speed of the stylus for the entire input path so that the calculated average speed of the stylus reflects its average speed when actually in motion.

[0128] In a similar line of reasoning, to the extent that the entered input path is longer and/or has a higher than average number of inflection points detected, then the input path can be proportionately less precise, since the more letters there are in the intended word the less likely the input path will closely match any other word. Thus, in another aspect, the amount by which the maximum matching distance thresholds are increased is a function of the total length of the input path. In another aspect, the amount by which the maximum matching distance thresholds are increased is a function of the number of inflection points identified in the input path. In another aspect, when the Pattern Matching Routine 3300 finishes searching the database of words and scoring metric for the most-closely-matching word falls below a determined threshold, then the maximum matching distance thresh-olds are increased, and the database of potentially match-ing words is searched again using these larger maximum matching distance thresholds.

[0129] While other forms of organization are possible for the database of words searched by the Pattern Matching Rou-tine 3300, in one aspect, the words in the database are organized in a fashion to facilitate efficient searching by the pattern matching component. Every input pattern has two easily and reliably identified inflection points - the first (PEN_DOWN) and the last (PEN_UP) - which, in one embodiment, are always unambiguously matched with the first and last letters, respectively, of the word being input. In one embodiment, the words in the database are orga-nized in a manner that takes advantage of this potential unambiguous matching to quickly and reliably eliminate a large number of potential candidate words from further consideration, thus shortening the time required to search the database. In one embodiment, words are stored in the database represented as strings of letters composed of an initial string component (hereinafter the “root” string) and a final string component (hereinafter the “suffix” string). Note that, although the term “suffix” is used to denote the final string component of a word, and while in many particular cases this component may in fact correspond to a suffix as it might be determined according to a linguistic analysis of a word according to the syntax of the natural language to which the word belongs, in the present invention the term “suffix” is used more generally to mean simply the final string component of a word representation in the database. Thus, for example, the suffix of a particular word representation may in fact be a “virtual” suffix in that all of the letters of the word are included in the root component, but wherein the representation of such a virtual suffix includes an indication of one or more of the final letters of the root portion so that the analysis described below can use the suffix in determining which words in the database are potentially matching candidate words with respect to an input pattern. Furthermore, the set of suffixes used within a given language database may be defined without regard to the identity of the set of suffixes syntactically determined by a linguistic analysis of the language. For example, the suffixes used may comprise the set of final letter-pairs that terminate one or more of the words in the database.

[0130] This approach of defining words in the database in terms of an initial root string and a final suffix string has two distinct advantages. One is that it economizes on storage, since each root and suffix string may be used to define a plurality of words in the database, but each root and suffix string need be stored only a single time in the database and referenced by means of an index each time it is used to define a word. A second advantage is that the PEN_DOWN inflection point and the initial portion of the input path can be analyzed to limit the number of possible matching roots to increase the efficiency with which possible word matches can be identified, since each root tends to be used in a plurality of words and the analysis of the root can be performed only once and need not be repeated for each word in which the root is referenced. Similarly, the PEN_UP inflection point and the final portion of the input path can be analyzed to limit the number of possible matching suffixes. To illustrate the great benefit of this approach, assume that each analysis of an input path, on average, eliminates all but 10% of the possible roots and 10% of the possible suffixes. Under the reasonable assumption that roots and suffixes are distributed more or less randomly among the words of the database, this means that, on average, only 1% of the database words needs to be analyzed in detail for each input path.

[0131] In another embodiment, each initial portion of a word is stored as an initial string (hereinafter referred to as a "prefix") followed by the remainder of the root portion (hereinafter the "root-core"). In another embodiment, all prefixes are defined to be of one fixed length so that each comprises the same number of letters. In another embodiment, each prefix is two letters in length, so that the initial letter pair of each word in the database is defined by a reference to the corresponding prefix. This approach has a number of benefits. One is that a table may be constructed with information regarding each letter-key pair that appears in a word of the database, either as an initial letter pair of a word or subsequently at a later position in a word. This table may be initialized with various data such as the distance between the centers of a letter-key pair and the slope of a directed line segment connecting the centers of the letter-key. Such information tends to be required repeatedly in the analysis of input patterns, and a great deal of computation can be saved by looking such information up in a table rather than calculating it each time. Thus, in addition to storing such information, a letter-key pair table can indicate where a group of words is stored where each word of the group begins with the indicated letter pair. Another advantage of this approach is that the set of initial letter-key pairs that can possibly match a given input path can be determined quickly and reliably.

[0132] In another embodiment, words are stored in the database as a paired reference to a root-core index and a reference to a suffix index. In another embodiment, a root-core may comprise a Null string of length zero. In another embodiment, a suffix may comprise a Null string of length zero that is associated with a single letter-key. This approach adds additional flexibility in defining root-cores and suffixes in a way that tends to minimize the total amount of storage space required by the database, while still allowing the identification of a set of potentially matching suffixes for an input path that can be used to limit the number of words actually analyzed in searching the database. In another embodiment, an indication of the frequency of use associated with a word is associated with its definition as a paired reference to a root-core index and a reference to a suffix index. In another embodiment, an indication of the total length of a representative input path for a word is associated with its definition. When the path length associated with a word differs from the length determined for an entered input path by more than a determined threshold difference, the word can be eliminated as a potentially matching candidate word without further analysis.

[0133] There are a number of methods that can be used to determine the set of roots (and, in the aspect mentioned above, prefixes) and suffixes that are actually used in representing the words of a language for a database. In one embodiment, in addition to including all single-letter suffixes and all "virtual" suffixes (as defined previously) that indicate the identity of the final letter-key of the preceding root, an analysis is performed to identify the most commonly occurring terminal strings including more than one letter among the words to be represented in the database. A number (for example, six hundred) of the most commonly occurring suffixes is chosen to use as an initial suffix set in analyzing the words to be included in the database (the “word list”). A first pass through the word list is performed, wherein each word is assigned the longest suffix in the suffix set that matches the final letters of the word. Since the suffix set includes all possible single-letter suffixes, every word is guaranteed to match at least one suffix. This first pass determines a set of roots that must be defined to represent every word in the word list when constrained to the initial suffix set, since the suffix referenced by a word then determines where the word must be “divided” between its root and suffix portion. The number of times each defined root and suffix is actually referenced by a word is determined. Then repeated passes are made through the word list wherein the words are analyzed again to determine whether all of the words that reference a given suffix (or root) can be divided at different locations within the words such that the total number of roots and suffixes required to represent all of the words in the word list can be reduced. In one aspect, the process begins by analyzing words which reference the least-frequently referenced roots and suffixes. The process is repeated until the algorithm determines that the number of roots and suffixes identified and the total amount of memory required to represent them cannot be further reduced. For example, in one embodiment, a figure of merit is determined that is a weighted function of the number of suffixes, the number of roots, and the total memory storage required. The weighting applied to the number of suffixes may be increased since the number of suffixes in the database affects the speed with which an input pattern can be analyzed. Similarly, the overall weighting applied to the number of roots or suffixes may be a non-linear function, since the number of each determines how many bits are required to store the reference to a root or suffix in the representation of a word in the final database. For example, if the total number of suffixes is constrained to be less than 1,024, the reference to a particular suffix can be stored in a field of only 10 bits. In another aspect, the definition of one or more suffixes references one or more functions which determine the final spelling of a word. For example, a suffix may be defined to begin with a function which has the effect of deleting the final letter of the preceding root. Thus, a database containing the words “defined,” “define,” and “defining,” a database might include the root “define” together with suffixes: “d”, (virtual) “e” and “[DELETE PREVIOUS CHARACTER]ing”. In combination, these yield:

"define" + (virtual) "e" = "define"; "define" + "d" = "defined"; and "define" + "[DELETE PREVIOUS CHARACTER]ing" = "defining". Another suffix function serves to repeat the immediately preceding character. Thus the word "preferred" could be defined in a database in terms of a root "prefer" and a suffix "[REPEAT PREVIOUS CHARACTER]ed". Such function-referencing suffixes reflect the morphological rules that determine how true linguistic suffixes commonly attach to word roots in a language. By incorporating such rules in the referenced suffixes of a database, the number of variant root forms required is significantly reduced, while the suffixes referencing such functions are no less useful in the processing and analysis of input patterns.

[0134] In addition, by incorporating an appropriate set of suffixes that reflect the morphological rules of a language, further reductions in the overall size of the database are possible. Many words exhibit the same morphological variation in a language, and groups of suffixes that can be defined such that a single word-entry can be created in the database that references such a group of suffixes to define all of the corresponding words in a single entry rather than multiple entries. For example, for many English adjectives, a set of suffixes including “ly”, “er” and “est” can be used to construct the adverbial, comparative and superlative forms, respectively. Thus, for example, a single entry referencing the root “great” and the above suffix group would generate “greatly”, “greater” and “greatest.” In another aspect, when words in the database include an indication of the frequency with which the word is expected to occur, such a suffix group includes a model of the relative frequencies generally expected for the various forms generated by the group with respect to the frequency of the most frequent form in the group. The frequency information stored in the database entry then indicates the frequency of the most frequent form, while the expected frequencies of the other forms are generated according to the frequency model associated with the group. In another aspect, when the expected frequency for a particular form of a particular word in such a word group differs from the frequency generated by the model by more than a threshold difference, a special entry (for example, consisting of a root combined with a single suffix) is created in the database which over-rides the group-generated form and its (incorrect) frequency information.

[0135] Returning to Figure 3C-1, at block 3325 a Suffix Candidate Identification Routine 33100 (as shown in Figure 3C-2) is called to determine which of the suffixes referenced in the database can be regarded as a potentially matching, and therefore valid, suffix with respect to the current input path. In accordance with another embodiment, at block 33110 the PEN_UP location is used to analyze and determine which of the set of possible suffixes referenced in the database can be considered as a possible match for the current input path by virtue of ending with a letter that is associated with a letter-key that is determined to be within the maximum matching threshold distance of the PEN_UP location. In a loop from block 33160 to block 33190, each potentially matching suffix is evaluated by a Suffix Scoring Metric Calculation Routine called at block 33180. For each suffix that is determined by the Suffix Scoring Metric Calculation Routine to conform closely enough to the input path, a numerical score is calculated that reflects how closely that suffix matches the input path. The Suffix Scoring Metric Calculation Routine is a modified version of the Scoring Metric Calculation Routine 3400, wherein in the modified version the letter-keys of a suffix are matched in reverse order starting from the end of the input path, and wherein no penalty is assessed when a suffix being scored is matched with only a final portion of the input path (as would be expected for any valid suffix). Since, as will be explained below, the exact score determined for the matching of a particular letter-key to a particular point on the input path is also a function of the matching of one or more adjacent letter-keys in a word, and since the same suffix can appear in different adjacent contexts in different words (depending on the final letter of the preceding root), for each scored suffix, the system preserves the specific (unadjusted) score for the first letter-key of each suffix (corresponding to the letter that immediately follows the root of a word) so that it can be adjusted in the context of each word in which it is used in order to determine a final score for that word. In another aspect (not shown in Figure 3D), since the root context of the first letter of a suffix is unknown at the time of its processing by the Suffix Scoring Metric Calculation Routine, depending on the nature of the input and the preceding root, it may not be possible to determine whether the first letter of a suffix should be matched with a particular inflection point or with a path segment adjacent to the inflection point. Where the matching of the first letter of a suffix is thus indeterminate, two provisional scores are determined and recorded for the first letter of the suffix, and the Scoring Metric Calculation Routine 3400 determines which provisional matching to use in scoring each particular word that references the suffix. If at block 33185 the Suffix Scoring Metric Calculation Routine determines that a suffix is a potentially valid suffix for the current input path, then at block 33187 the suffix is temporarily marked in the database as valid. When all potentially matching suffixed have been examined by the Suffix Scoring Metric Calculation Routine, at block 33130 the Suffix Candidate Identification Routine terminates and returns to block 3330 in Figure 3C-1.

[0136] At block 3330, in another embodiment, the process identifies each unique ordered pair of keys such that the first key of each pair is located within a predetermined maximum matching threshold distance from the PEN_DOWN inflection point, and the second key of each pair is located within a predetermined maximum matching threshold distance from the next detected inflection point location or is within a predetermined maximum matching threshold distance from the 1st or 2nd path segment (preceding or following the next detected inflection point location). This set of ordered pairs of letter-keys completely determines a set of two-letter prefixes wherein any potentially matching word in the database must reference one of the prefixes in the set, so that all other words in the database that reference prefixes not included in the set can be skipped in the database searching process.

[0137] Then in another embodiment, at block 3335, the process identifies the range of input path lengths that are associated with words in the database that are considered to be potential matches with the input pattern based on its actual length as measured by the Input Pattern Analysis Routine and stored in the IPT. Then the loop from block 3340 to block 3350 is executed for each ordered pair of keys identified at block 3330, where at block 3345 the group of words in the database is identified that references any prefix that corresponds to the current ordered pair of letter-keys. Then the loop from block 3360 to block 3390 is executed for each word in the group of words identified at block 3345. At blocks 3365 and 3370, the process checks whether the word qualifies as a candidate based on the suffix referenced by the word and on the input path length (or path length class) associated with the word. In one embodiment, if either of these qualifications is not met, then the word is skipped over and the next word is considered. In another embodiment (not shown in Figure 3C-1), each word in the database is also stored with an expected minimum number of inflection points based on the geometric two-dimensional relationship between the letter-keys associated with the sequence of letters that form the word's spelling. A candidate word does not qualify if fewer than the expected minimum number of inflection points are identified in the current input pattern. If the word meets all of the preliminary qualifications, then at block 3375 the sequence of letter-keys associated with the word is copied to an array KEYS[], the variable SUFFIX is set to the index value that identifies the suffix referenced by the current word, and WORD_Len is set to its length. At block 3380, a Scoring Metric Calculation Routine 3400 (shown in Figure 3D) is called to calculate the Scoring Metric value for the current word. Then at block 3385, an Update Word_Candidates Table Routine 3700 (shown in Figure 3G) is called to determine whether the calculated Scoring Metric value is good enough to qualify the current word to be provisionally added to the Word_Candidates[] table of the top matching word candidates identified in the database, and if so, to add it to the table. Once all of the identified qualifying candidate words have been evaluated, the routine returns at block 3355 to the Main Processing Routine 3100, where at block 3125 the Display Selection List Routine 31000 (shown in Figure 3J) is called to determine whether to immediately generate the top-scoring Word Candidate for output as the textual interpretation of the input pattern, or to display the identified word candidates in a selection list on the text display 2106.

[0138] Figure 3D-1 shows a flowchart of an embodiment of the Scoring Metric Calculation Routine 3400, called at block 3380 of the Pattern Matching Routine 3300. The process depicted in Figure 3D-1 determines whether it is possible to identify a valid matching between the inflection points (and, if necessary, path segments) of an input pattern and the keys associated with the letters of a candidate word. If such a matching is possible, the routine of Figure 3D-1 identifies an optimal or close to optimal matching so that a Set Scoring Metric Value Routine 3700 called at block 3490 can quickly and simply calculate the actual value of the Scoring Metric according to the identified matching arrangement. Note that the algorithm of Figure 3D-1 does not require the use of recursion, and is extremely fast and efficient. An implementation of the present invention runs in real-time on a standard Hewlett-Packard iPAQ 2215 PocketPC device with an Intel XScale 400 MHz processor, completely processing each input path and completing a search of a 60,000 word database with an average response time of less than 150 milliseconds from the receipt of the PEN_UP data point.

[0139] The algorithm shown in Figure 3D-1 matches the letter-keys corresponding to a candidate word of length WORD_Len (stored in an array KEYS[]) with the set of inflection points determined for an input pattern and stored in the IPT. In one embodiment (not shown in Figure 3D), as the inflection points and path segments of the current input path are matched in sequence with the letter-keys of a candidate word, when the next letter-key of a candidate word is greater than the determined maximum matching threshold distance from both the current path segment and the next inflection point to be matched, the current word is rejected as a possible candidate to match the current input pattern and the system proceeds to evaluate the next possibly matching candidate word. In another embodiment (also not shown in Figure 3D), for computational efficiency, the Scoring Metric Calculation Routine makes use of the result obtained from the original evaluation of the suffix associated with a word (performed by the Suffix Scoring Metric Calculation Routine to determine whether words referencing a particular suffix are to be considered as potentially matching candidates) and calculates a Scoring Metric result only for the root portion of each word, merging that result with that obtained for the suffix to determine a final Scoring Metric value for each complete word. For simplicity, the embodiment shown in Figure 3D depicts a method wherein a Scoring Metric value is calculated based on analyzing all of the letter-keys corresponding to the entire candidate word.

[0140] In another aspect, when the distance from a determined inflection point of the input path to the next corresponding letter-key of a word is greater than the determined maximum matching threshold distance, the word is retained as a word candidate and a scoring penalty is applied to the word that is used in computing the scoring metric for the word (hereinafter, the "skipping penalty" since the inflection point is "skipped" in the matching process for the word). In one aspect, different skipping penalties are assessed for different types of inflection points, where higher penalties are assessed for skipping an inflection point of a type that is deemed more likely to correspond to a letter of the intended word. In another aspect, other skipping penalties are also assessed when an inflection point is only partially matched, such as a DOUBLE_LETTER inflection point that is matched with a letter-key corresponding to a letter in a word that is not repeated, or a DOUBLE_LETTER inflection point that is matched with two neighboring, but distinct letter-keys that correspond to two different successive letters of a word. Although not shown in block 3330 of Figure 3C-1 (for simplicity), in another embodiment, additional prefix letter-key pairs are identified in which either or both of the first two detected inflection points are skipped, and wherein the corresponding skipping penalties are determined and assessed by the Scoring Metric Calculation Routine of Figure 3D against every word referencing such additional identified prefix letter-key pairs. Similarly, in another embodiment, additional prefix letter-key pairs are identified in which the letter-keys of the pair are matched with the first two detected inflection points in reverse order (corresponding to a spelling transposition error as described below), and wherein the corresponding letter-transposition penalties are determined and assessed by the Scoring Metric Calculation Routine against every word referencing such additional identified prefix letter-key pairs.

[0141] In another aspect, an additive skipping penalty is assessed against the word being scored by adding a determined weighted penalty amount to the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched in calculating the scoring metric for the word. In another aspect, the additive skipping penalty is determined to be the maximum matching threshold distance determined for the type of inflection point that is skipped, wherein the standard weighting applied for the type of inflection point that is skipped is used to weight this additive penalty amount in calculating the scoring metric for the word. In another aspect, the additive skipping penalty is determined to be the average distance determined for each letter-key of the word that is matched to a corresponding point of the input path, wherein the standard weighting applied for the type of inflection point that is skipped is used to weight this additive penalty amount in calculating the scoring metric for the word.

[0142] Another aspect relates to the embodiment in which a small circular gesture is performed to create a DOU-

BLE_LETTER inflection point. Depending on the relationship between the direction from which the double-letter key is entered and the direction in which the input path continues when leaving the key, it may be difficult for a user to determine how many repetitions of a “small circular gesture” have actually been performed. In yet another aspect, additional repetitions of the movement are processed by the system in the same fashion as a single performance of the movement. In this aspect, one or more repetitions of the movement are matched with any sequence comprising two or more consecutive occurrences of the same letter in a word, or two or more consecutive occurrences of alternative forms of the same letter. In yet another aspect, when a small loop is detected in the input pattern that does not complete a full 360° change in the slope of the input path, an alternate class of DOUBLE_LETTER2 inflection point is identified that is matched by the system both as an ANGLE_THRESHOLD inflection point and as a true DOUBLE_LETTER inflection point, choosing the appropriate interpretation for each evaluated candidate word that best matches the word. An example of such an inflection point can be found in entering the input paths for the words “fed” and “feed,” where there may be ambiguity in whether the traced path was intended to contain a double letter gesture. The path may begin within the “f” key, proceed up and over to the “e” key, and depending on the user's habits in tracing an input path, the path may inadvertently loop over to the right and back down across itself in proceeding down to the “d” key. Assuming the path traverses the center of the “f” and “d” keys and loops around the center of the “e” key (an ideal path for correct recognition), the slope of the path changes by only about 220° between the entry into and exit from the “e” key. At the same time, this path may seem like a perfectly correct execution of a DOUBLE_LETTER gesture to a user who intends to enter the word “feed”, since a small closed loop has been traced on the “e” key. In another aspect, a separate additive penalty factor is calculated for use with a DOUBLE_LETTER2 inflection point when matched with a single vs. a double letter. Two flags determine whether an adjustment factor is added to the scoring metric calculated for a candidate word when a DOUBLE_LETTER2 inflection point is matched with a single letter, or when it is matched with a double letter. When a small loop is traced by the user that does not complete a full 360° change in slope, and the word choice list contains words which match for both a single letter and a double letter, the system determines which of the two alternate words was selected for output by the user. When the user's selection appears lower in the selection list than the corresponding alternate word, an incremental change is made in the adjustment factors (and when necessary, the associated flag values) used for a DOUBLE_LETTER2 inflection point. This enables the system to better accommodate a user who habitually creates small loops at inflection points where the path “reverses” direction in both the x- and y- components, as in the current example. In another aspect, separate values of the flag and adjustment factors are calculated for different ranges in the number of degrees by which the slope changes in completing the loop gesture.

[0143] In another aspect, a multiplicative skipping penalty is assessed against a word being scored by increasing a determined multiplicative penalty factor by which the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched is multiplied when calculating the scoring metric for the word. In one aspect, the amount by which this multiplicative skipping penalty factor is increased is determined with respect to the type of inflection point that is skipped. In another aspect, when the multiplicative penalty factor determined for a word exceeds a determined threshold, then the word is eliminated as a potentially matching candidate for the input pattern. In another aspect, in calculating the scoring metric for a word, a separate multiplicative factor is applied to the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched, where this separate multiplicative factor is determined with respect to the frequency associated with the word being scored, and wherein this multiplicative factor is larger for words associated with lower frequencies.

[0144] The skipping penalty approach described above provides a computationally simple approach that achieves a surprising and useful result. This method of determining the scoring metric for words enables the system in a significant number of cases to correctly identify the intended word despite significant errors on the part of the user in entering the input pattern. One type of error that is frequently accommodated is the deliberate insertion of an additional letter in the course of entering an input pattern corresponding to the misspelling of the intended word. In this case, the system produces the correctly spelled version of the intended word (assuming that the misspelled version of the word has not been deliberately added to the database). Another type of error that is frequently accommodated is where the user has become lost, distracted, or confused and begins to trace out the input path in a direction that does not lead toward the next letter of the word, and in realizing and correcting the mistake, a distinct ANGLE_THRESHOLD inflection point is generally created. Since an average user will frequently pause at such a time, a PAUSE_ANGLE inflection point (as described above) is often detected. Since the situation just described is often the reason why such a PAUSE_ANGLE inflection point is created, inflection points of this type are associated with both a much lower-than-average weighting and a very low skipping penalty.

[0145] As was just described, the skipping penalty approach for inflection points detected in the input pattern that are skipped in the process of matching the letter-keys of a word allows for the generation of a correctly spelled version of an intended word despite the presence of spelling letter-insertion errors on the part of the user. With a large database of words for a language (for example, over

60,000 words for United States English), there tends to be a high “density” of input patterns corresponding to words in the database that are relatively short. In such cases, errors of this type often produce input patterns that are determined to more closely match one or more words other than the intended word. However, even in such cases, the intended word frequently appears in the list of alternate word choices presented to the user. When the intended word is a longer word, the intended word is frequently determined to be the most closely-matching word. This approach is consequently very successful in accommodating spelling insertion errors and unintended path entry errors.

[0146] In a similar but distinct aspect, skipping penalties are assessed when a letter-key of a word is greater than the determined maximum matching threshold distance from any potentially corresponding point of the input path (hereinafter, a “letter-skipping penalty”). This corresponds to a letter-omission spelling error in tracing out the input path for an intended word. It is less likely to correspond to a simple path-entry error, since the present invention allows for the use of very liberal maximum distance thresholds, such that even quite sloppy attempts to enter a given letter will usually be able to be matched to the intended letter. In another aspect, a letter-transposition penalty is also determined and assessed when two letter-keys of a word are matched to corresponding points of the input path that occur in reverse order on the input path as compared to order of the letters in the correct spelling of the word. This corresponds to a letter-transposition spelling error in tracing out the input path for an intended word. In another aspect, when it is determined that both an inflection point skipping penalty and a letter-skipping penalty should be assessed against a word being scored, and both penalties are detected in the analysis between the matching with the input path of the letter preceding the skipped letter and of the letter following the skipped letter (or when the skipped letter and the skipped inflection point both occur at the beginning of the word and input path, respectively, or both occur at the end of the word and input path), then a single letter-substitution penalty is assessed against the word being scored. This is reasonable since a letter-omission error and a letter-insertion error that both occur at the same point within a word is equivalent to substituting one letter for another in the spelling of a word.

[0147] In an aspect analogous to that for inflection point skipping penalties, an additive letter-skipping penalty (or, in the case of a letter substitution, an additive letter-substitution penalty) is assessed against the word being scored by adding a determined weighted penalty amount to the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched in calculating the scoring metric for the word. In another aspect, the additive letter-skipping penalty (or letter-substitution penalty) is determined to be the average distance determined for each letter-key of the word that is matched to a corresponding point of the input path, wherein a standard determined weighting for skipped letters is used to weight this additive penalty amount in calculating the scoring metric for the word. In assessing a letter-transposition penalty, no additive penalty is assessed since each letter-key of the word is still matched with a corresponding point of the input path from which an appropriately weighted distance is measured.

[0148] In another aspect, a multiplicative letter-skipping penalty (or, in the case of a letter substitution, a multiplicative letter-substitution penalty) is assessed against the word being scored by increasing a determined multiplicative penalty factor by which the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched is multiplied when calculating the scoring metric for the word. In another aspect, a multiplicative letter-transposition penalty is assessed against the word being scored by increasing a determined multiplicative penalty factor by which the sum of the weighted distances of the letter-keys of the word from the path points with which they are matched is multiplied when calculating the scoring metric for the word.

[0149] Turning to Figure 3D-1, at block 3405, separate variables keyIndex and ipIndex are initialized to 0 so that they can independently step through the letter-keys of the word, and the inflection points and path segments of the input path, respectively, in order to determine an appropriate matching. Loop limits keyLimit and ipLimit are set to the number of the letter-keys in the current word and the number of inflection points in the IPT, respectively, and the variables used to accumulate the various coring components are initialized to zero. Then at blocks 3410 and 3415, the main loop limits are tested, and the process exits from the main loop to block 3470 as soon as either limit is reached.

[0150] At block 3420, for each letter-key of the word to be matched to the input path, the process identifies the preceding, current and following letter-keys as lastKEY, KEY and nextKEY, respectively, where lastKEY and nextKEY are set to recognized invalid values at blocks 3405 and 3430 to address the boundary conditions where there is no preceding or following letter-key. The next inflection point in the IPT to be matched is accessed through the variable IP, and matchFlag is initialized to an invalid value. Then at block 3435 the Letter-Key to Path Matching Routine shown in Figure 3D-2 is called to identify whether and where the current letter-key KEY should be matched to the input path.

[0151] Then in Figure 3D-2 at block 34105, the process determines whether KEY is within a maximum matching threshold distance from the inflection point IP. If so, at block 34110, the process then determines whether nextKEY is also within a maximum matching threshold distance from the inflection point IP. If so, the process begins at block 34115 to determine which letter-key should be matched with IP by first checking whether they are actually the same letter-key repeated twice in sequence (reflecting a repeated letter in the candidate word). If they are the same key, then at block 34120 the process checks whether the current IP is a type DOUBLE_LETTER inflection point, and if so, at block 34125 matchFlag is set to reflect the matching of a double-letter with a DOUBLE_LETTER inflection point. If the current IP is not a DOUBLE_LETTER inflection point, then at block 34175 the process determines whether there is a DOUBLE_LETTER inflection point that is coming up soon enough on the input path that the matching of KEY and nextKEY should be delayed until the upcoming DOUBLE_LETTER inflection point is reached, and if so, at block 34180 the process returns with matchFlag still set to an invalid value so that the match is delayed until the appropriate time. If not, at block 34125 matchFlag is set to reflect the matching of a double-letter with an inflection point that is not of type DOUBLE_LETTER. If on the other hand, at block 34115 KEY and nextKEY are found to be different, and at block 34150 the current IP is in fact a DOUBLE_LETTER inflection point, then at block 34155 the process determines whether there is a double-letter that is coming up in the candidate word and that all preceding letter-keys can be successfully matched with the current path segment, and if so, at block 34165 the process returns with matchFlag set at block 34160 so that that KEY will be matched with the current path segment. If at block 34150 the current IP is not a DOUBLE_LETTER inflection point, then at block 34170 the process determines which matching of KEY and nextKEY (with respect to the current IP, the current and following path segments, and following inflection point) minimizes the total score assessed for the word and sets matchFlag accordingly before proceeding.

[0152] At block 34130, the process has determined the location on the path with which KEY is to be matched, and, since the letter-keys of a word are matched in sequence, the process has also determined the location on the path with which the preceding letter-key is to be matched. In another related aspect, in calculating the scoring metric for a word, for each successive letter of the word, the distance of the letter-key from the path point with which it is matched is adjusted by a factor that is a function of the ratio (or, in an alternate embodiment, the difference) between the actual path length and the straight line distance between the matched path point and the path point matched with the preceding letter. This approach provides a useful result because the more the input path deviates significantly from a straight path between two successively matched points, the more likely it is that the path diverts to pass near a letter-key that is part of the user's intended word, but that is not part of the word being scored. This approach thus tends to penalize the scores of words other than the user's intended word. In another aspect, the amount of influence of this factor is scaled depending on the separation between the two successive matched points, increasing the effect of the factor for keys that are more widely separated, and reducing it for keys that are in closer proximity (and eliminating it as a factor when keys are closer than a determined minimum threshold). This avoids attributing an inappropriately large effect on the scoring metric to a path deviation that is not really significant, since it is common for an input path to deviate noticeably from a straight-line path between keys, and even a slight deviation over a relatively short distance can result in a path-length to straight-line ratio that would have a significant effect on scoring. Thus, at block 34130, a PathLengthFactor is calculated in accordance with the above embodiment.

[0153] In a related but distinct aspect, in calculating the scoring metric for a word, for each successive letter of the word that is matched with a detected inflection point, the distance of the letter-key from the inflection point with which it is matched is adjusted by a factor that is a function of the ratio between the straight line distance between the matched inflection point and the preceding inflection point, and the straight line distance between the letter keys of the word being scored that are matched with these inflection points. The amount by which the letter-key-to-inflection-point distance is adjusted is a function how much the ratio of the two straight-line distances varies from a 1-to-1 ratio. This approach provides a useful result because, on average over the course of scoring a word, this ratio should tend to closely approximate a 1-to-1 ratio for each letter of the user's actual intended word, while it will generally tend to diverge from a 1-to-1 ratio for other words. This approach thus also tends to penalize the scores of words other than the user's intended word. Thus, at block 34130, a LengthFactor is also calculated in accordance with this embodiment.

[0154] In another related aspect, in calculating the scoring metric for a word, for each successive letter of the word, the distance of the letter-key from the path point with which it is matched is adjusted by a factor that is a function of the difference between the slope of a line between the matched path point and the path point matched with the preceding letter, and the slope of a line between the two corresponding matched letter-keys. This approach provides a useful result because, on average, as the user traces out an input path for a word, the slope of each path segment between successive letters should approximate the slope between the letters themselves. In contrast, for words other than the intended word, there will generally be greater deviations between these slopes. This approach thus also tends to penalize the scores of words other than the user's intended word. In another aspect, the amount of influence of this factor is scaled depending on the separation between the two successive matched points, increasing the effect of the factor for keys that are more widely separated, and reducing it for keys that are in closer proximity (and eliminating it as a factor when keys are closer than a determined minimum threshold). This avoids attributing an inappropriately large effect on the scoring metric to a slope deviation that is not really significant, since, when two keys are relatively close, it is possible for an input path to pass close by both keys without having to significantly change direction in moving from one to the next. In another aspect, when the difference between the two slopes exceeds a determined threshold, then it is determined that the letter-key cannot validly match the path at the originally matched point. In another aspect, when a letter-key has been matched with an inflection point and the two slopes exceed the determined threshold, then an alternative matching path point is determined as the closest path point to the letter-key, and the slope difference is re-calculated based on the alternative matching path point. If the slope difference calculated for the alternative matching path point location is less than the determined threshold, then the calculated slope difference is used to adjust the distance of the letter-key from the originally matched inflection point. This alternative matching procedure accommodates cases where, for example, the user "overshoots" a letter-key in a direction that is essentially perpendicular to the line between the two matched letter keys. In such cases, the input path will still tend to pass close enough to the letter-key of the intended word (before "overshooting") that the closest point on the input path will still pass the slope-difference test. This adaptation ensures that the word actually intended by the user does not fail the test. In contrast there is no serious detriment to the performance of the system when an unintended word passes this test when it should actually fail, since the combined effect of the other aspects of the present invention make it highly unlikely that such an unintended will result in the best numerical scoring. Such pass/fail tests must in general be quite conservative to avoid the serious error of disqualifying the intended word. Thus, at block 34130, a SlopeFactor is also calculated in accordance with this embodiment, and at block 34135 the calculated SlopeFactor is compared to a maximum threshold that is determined with respect to the separation between the two matched path points.

[0155] If at block 34135 the calculated SlopeFactor exceeds the determined threshold, then the provisional matching determined between KEY and the current inflection point is invalidated and, at block 34140, the process determines if KEY is within a determined maximum matching threshold distance of a determined matching location on the current path segment. If so, then in accordance with the above disclosed embodiments, at block 34185 new values of SlopeFactor, LengthFactor and PathLengthFactor are calculated with respect to the new provisional matching location, and at block 34190 the new value of SlopeFactor is compared to a determined maximum threshold. If at block 34140, the process determines that KEY is not within the determined maximum matching threshold distance, or if at block 34190 the process determines that SlopeFactor exceeds the determined maximum threshold, then at block 34145 the process returns with matchFlag still set to an invalid value so that the attempt to match KEY to the input path fails in this path region. If at block 34135 or block 34190 the calculated value of SlopeFactor falls below the determined maximum threshold, then, in one embodiment, at block 34195 the calculated values of SlopeFactor, LengthFactor and PathLengthFactor are used in calculating an adjustment to the weighting that is applied to the calculated distance, where this weighting is summed with all other determined weightings (in the variable weightSum at block 3430 of Figure 3D-1). In another embodiment, at block 34195 the calculated values of SlopeFactor, LengthFactor and PathLengthFactor are used in calculating an adjustment to the calculated distance itself, prior to applying any weighting. Note that these factors are used in such a way that lower values of SlopeFactor, LengthFactor and PathLengthFactor (that tend to indicate that the key is appropriately matched at the determined location) result in lower calculated distances and higher weightings. At block 34199, the process returns all of the calculated values to the calling routine at block 3425 of Figure 3D-1.

[0156] At block 3425, if matchFlag is set to indicate that that a valid match was established for KEY, then at block 3430 the calculated values are summed into rawDistance, rawScore and weightSum, and the type and location of the match is recorded. Then, in another embodiment, at block 3433, if the preceding key was flagged as skipped, the Letter-Key to Path Matching Routine is called to determine if the it can now be matched to the input path (at a location following, rather than preceding, the letter-key just matched), corresponding to a letter-transposition spelling error in tracing out the input path for the intended word. If so, and if the following letter is also successfully matched, then a letter-transposition is assessed for the current matching. In another embodiment, if at block 3425, matchFlag is set to indicate that that no valid match was established for KEY, then at block 3435 the process determines whether KEY can subsequently be successfully matched with the input path at a location prior to a location where the following letter-key can also be matched. If not, then the process determines that KEY cannot be matched to the input path and must be skipped, so that at block 3440, skippedKeys is incremented and a determined penalty is added to keyPenalty. If at block 3435 the process determines that KEY can still be successfully matched at a later point in the input path, then at block 3445 keyIndex is decremented so that the same key is processed in the next loop iteration at block 3420. Then, since it can be inferred from this result that the current inflection point (IP) must be skipped, the process proceeds to block 3460, where skippedIPs is incremented and a determined penalty is added to ipPenalty. Alternatively, if at block 3440 the current IP has been flagged as skipped as a result of the manner in which KEY was matched to the input path, the process also continues at block 3460. Then at block 3465, ipIndex is incremented to skip over the current inflection point in the next loop iteration at block 3420. If however, at block 3440 the current IP has not been flagged as skipped, then at block 3455 and 3465, ipIndex is incremented only if the current IP has been matched to a letter-key.

[0157] When, at block 3410 or 3415, either of the main loop limits are reached, the process exits from the main loop to block 3470 and determines whether any letter-keys or inflection points remain yet unmatched, and if so, at block 3475 corresponding additional skipping penalties are assessed. Then, in another embodiment, at block 3480 the process determines whether the assessed penalties exceed a determined maximum threshold, and if so, at block 3485 the process returns a flag value that the current candidate word is not a valid match for the input path. Otherwise, at block 3490 a Set Scoring Metric Value routine (shown in Figure 3E) is called to calculate the actual Scoring Metric value for the current candidate word, based on the determined values for rawScore, weightSum, assessed penalties, and other data associated with the candidate word such as frequency information. Then at block 3495 the process returns a flag value that the current candidate word is a valid match for the input path, along with the determined Scoring Metric value.

[0158] In another embodiment, the algorithm of Figure 3D is modified so that it alternates between matching the next inflection point moving forward from the first PEN_DOWN inflection point, and moving backward from the last PEN_UP inflection point. In yet another embodiment, any identified DOUBLE_LETTER inflection point (or points) is first matched to any occurrence (or multiple occurrences) of a double letter in the candidate word, and the algorithm of Figure 3D alternately moves forward and backward from each of the a priori matched PEN_DOWN, PEN_UP, and DOUBLE_LETTER inflection points.

[0159] Once the Scoring Metric Calculation Routine 3400 has identified a valid pairing between the identified inflection points and the letters of a candidate word, the actual numeric value of the Scoring Metric is determined in the Set Scoring Metric Value Routine 3500 shown in Figure 3E (called from block 3490 in the Scoring Metric Calculation Routine 3400). At block 3505, avScore is set to the average weighted score per letter-key, omitting any skipped keys. Similarly, avDistance is set to the average unweighted distance as measured to each letter-key matched to an input path point. In one embodiment, an additive scoring penalty is assessed by setting skipCount to either skippedKeys or skippedIPs, whichever is greater. In another embodiment, an additive scoring penalty is assessed by setting skipCount to the sum of skippedKeys and skippedIPs. Then rawScore is increased by the quantity (skipCount * avScore). In another embodiment, rawScore is incremented by a fixed penalty amount at blocks 3440 and 3460 of Figure 3D-1 each time a key or inflection point is skipped, where the fixed penalty amount is determined with respect to the type of inflection point (or path point) that is skipped.

[0160] Similarly, an additive distance penalty is assessed by adding to rawDistance a fixed MaxDistance penalty amount for each skipped letter-key. In another embodiment, rawDistance is incremented by a fixed penalty amount at block 3440 of Figure 3D-1 each time a key is skipped, where the fixed penalty amount is determined with respect to the type of inflection point (or path point) that is skipped. In one embodiment, the average unweighted distance to each letter-key (avDistance) is then calculated as the incremented value of rawDistance divided by the number of letters in the word.

[0161] In another embodiment, the additive distance penalty is assessed by adding a fixed penalty amount to rawDistance for each skipped inflection point, where the penalty amount is determined with respect to the type of inflection point that is skipped. In another embodiment, the average distance avDistance is then calculated by dividing rawDistance by a quantity that is determined as the number of letters in the word minus the number of skipped keys plus the number of skipped inflection points. As will be seen in the Display Selection List Routine of Figure 3J, in another embodiment, the value of avDistance for each candidate word is used as a secondary scoring metric in determining how to present the list of possible candidate words.

[0162] Then at block 3510, a weighted score (weightedScore) is calculated by dividing the value of rawScore (including any additive penalties) by weightSum, the cumulative total of the weightings assigned to the matching of each letter-key by the Scoring Metric Calculation Routine at block 34195, where each weighting is a function of the characteristics of the path point with which the letter-key is matched and, in another embodiment, the relationship with a preceding (or following) letter-key and the path point with which it is matched. Then at block 3515, in another embodiment, any multiplicative penalty that has been assessed is applied in calculating the value penalizedScore by multiplying the value weightedScore by the sum of 1 plus any multiplicative penalties summed in ipPenalty and keyPenalty. Then at block 3520, in another embodiment, a value WordFrequencyWeightingFactor is calculated as a function of the average pen speed measured for the current input path and the historical average and variance of the pen speed, wherein the default value of a weighting factor (determined with respect to the frequency associated with the word) is adjusted with respect to the relationship between the pen speed measured for the current input path and the historical average and variance of pen speeds measured for previously entered paths. In one embodiment, the calculated WordFrequencyWeightingFactor is adjusted to a minimum value when the pen speed for the current input path is less than or equal to a value three standard deviations below the historical average, and is adjusted to a maximum value when the pen speed for the current input path is greater than or equal to a value three standard deviations above the historical average, and is adjusted proportionally in between these extremes. In another embodiment, the default frequency weighting factor (prior to any adjustment) is calculated as:

[0163] (1 + log(MAX_FREQ / WORD_Frequency))

[0164] where MAX_FREQ is the maximum possible frequency of use value that can be associated with a word in the database (for example, for the English language the frequency of the word “the”) and where WORD_Frequency is the frequency value associated with the current word. Then at block 3525, a Scoring Metric value is calculated by multiplying the penalizedScore by the calculated WordFrequencyWeightingFactor.

[0165] In another aspect, the method allows the user to indicate the desired case for any letter of a word being input. In one embodiment, the displayed keyboard includes a Shift function key, and the Shift key is tapped prior to entering an input pattern to shift the first letter of the word to upper case in the text word that is output or that is offered to the user for selection. Since an input pattern for a word must be entered as a continuous stroke, it is not possible to lift the stylus to tap the Shift key for a letter other than the first letter of a word (such as the upper-case `D' in the word “McDonald”). In another aspect, when the system detects an inflection point within a determined threshold distance from the Shift key, and the detected inflection point is not otherwise matched with a letter-key of a candidate word, then the next letter of the candidate word that is matched with a subsequent point of the input path is shifted to upper case in the text word that is output or that is offered to the user for selection. In yet another aspect, the user may indicate that a letter of the intended word is to be shifted to upper case by moving the stylus in a distinct gesture immediately before, during, or immediately after moving the stylus to the vicinity of the letter-key corresponding to the letter to be shifted. In one embodiment, after moving to the vicinity of the letter-key corresponding to the letter to be shifted, the stylus is moved upward above the upper border of the virtual keyboard and returned to the keyboard. In another aspect, when the stylus makes such a “shift-gesture excursion” above the top of the virtual keyboard and returned, and a letter-key can be matched with either the segment immediately preceding the excursion or the segment immediately following the excursion, then the letter-key is matched with the segment immediately following the excursion. This approach has been found to enable the user to naturally and reliably shift any intended letter of a word. In another aspect, when a candidate word is determined which has a letter other than the first letter shifted to upper case, and where the first letter itself is not shifted to upper case (hereinafter a “shift anomaly,” since in general usage words are not entered in such a fashion), then irrespective of the scoring metric determined for the word, the word is demoted in any word selection list below all other words that do not exhibit a shift anomaly. For example, Figure 5 shows an input path 5100 in which a shift-gesture is used to capitalize the first letter of the word “Dan.” To enter an input path corresponding to the capitalized word “Dan”, the stylus would first make contact at a PEN_DOWN location 5102 near the letter-key for `d', then move upward above the top of the keyboard (exiting from the defined keyboard region at a point 5104) in performing the shift-gesture before re-entering the virtual keyboard to move to the vicinity of the letter-key for `a' and then turning to move to the vicinity of the letter-key for `n' before lifting the stylus at the end of the input path at a point 5106. In performing the shift-gesture, it is likely that the stylus would pass extremely close to the center of the letter-key for `e' (as shown, at a point 5108), so that it is not unlikely that the word “dean” could actually receive a better scoring metric result than the intended word “Dan” so that “dean” might otherwise be calculated to be the most likely matching word. However, due to the relation of the shift-gesture to the locations of the letter-keys for the word “dean,” the actual text determined would be “dEan” so that, the shift anomaly would result in its being demoted below the intended word “Dan.” In a like manner, if in fact the user's intention were to enter the capitalized word “Dean” then on re-entering the virtual keyboard, the input path would again pass near the letter-key for `e' since this would be the next target letter for the user following the initial capitalized `D' and, in accordance with the aspect disclosed above, the `e' in “Dean” would be associated with the segment following the shift-gesture excursion.

[0166] Returning to Figure 3E, in accordance with this embodiment, at block 3530, for each identified shift gesture location (corresponding to location 5104 in Figure 5), the process identifies the word letter for which the matched input path point is closest to the detected shift gesture location along the preceding portion of the input path. Each such corresponding letter is then shifted to upper case before presenting the text of the word to the user for selection or outputting the word as generated text corresponding to the input path. In another embodiment, at block 3535 the process determines if the first letter of the word has not been shifted but one or more following letters have been, then at block 3545 the word is flagged as containing a shift anomaly. Otherwise, at block 3540 the shiftAnomaly flag is cleared before returning the calculated values at block 3550.

[0167] In another aspect (not shown in Figure 3E or Figure 5), the method recognizes a gesture indicating that each letter of the word determined for the input path currently being entered should be generated for output in an upper case form. At anytime between the initial contact of the stylus with the keyboard near the first letter-key of the intended word and the final contact location of the stylus near the last letter-key of the intended word, the user moves the stylus off of the defined virtual keyboard area without breaking contact between the stylus and the touch-screen and performs a determined gesture before moving the stylus back onto the defined virtual keyboard area. In one preferred embodiment, the determined gesture comprises moving the stylus in a closed loop such that the direction of movement turns through at least 360 degrees before re-entering the defined virtual keyboard area. This embodiment is very useful, since in many cases where a "Caps-Lock" type function is used, it is used only for a single word (commonly an acronym) rather than for a number of words. This embodiment enables the user to conveniently shift all of the letters of a single word to upper case with a single gesture that can be performed at any point in entering the word rather than having to explicitly activate a Caps-Lock function and then de-activate it after entering the word.

[0168] Figure 3F shows an embodiment of the Process TAP Inflection Point Routine 3600, called from the Pattern Matching Routine 3300 at block 3315. At block 3603, the process determines whether the TAP location occurred within the boundaries of a key that is associated with the generation of a letter. If so, then at block 3605 the process checks whether a tapped word has already been started, or whether this is the first tap of a new sequence of one or more taps (i.e. TAP_Word_Len is currently set to 0).

[0169] Many commonly used languages make extensive use of characters with diacritic markings. In one previously disclosed embodiment, words are stored in the database with their correct diacritic spellings, so that various input paths may be entered for different words making use of a single virtual key to refer to more than one diacritic form of a letter, where each of the diacritic forms of a letter are associated with a single virtual key. In another previously disclosed embodiment (shown at block 3225 of Figure

3B-1) the present invention facilitates the explicit entry of alternate diacritic forms of a character by associating the various alternate forms of a character with a single virtual key and displaying a selection menu enabling the user to select a desired alternate form upon detecting that the stylus has been held in contact with the virtual key for longer than a determined threshold period of time. These two embodiments eliminate the need to change to alter-nate keyboard layouts in order to access alternate diacritic forms of characters. A further observation is that in many cases, the average frequency of occurrence of various dia-critic forms of a given letter in a given language differ based upon the length of a word in which the character appears, or alternatively, upon the relative position of the character within the word. For example, in the French lan-guage, various diacritic forms of the letter “a” are used, including frequent use of both “a” (with no diacritic marks) and “à” (with an accent grave). However, although the form “a” is used not infrequently as a one-letter word, the one-letter word “à” has a much higher frequency of oc-currence. However, the form “a” is used far more fre-quently than the form “à” as an internal letter within a multi-character word. Therefore, when tapping a sequence of one or more keys to generate one letter per key-tap, with respect to a key with which more than one diacritic form of a letter is associated, it would be advantageous to define more than one default letter-form associated with the key, such that the context in which a particular key-tap occurs determines which default form is generated. For example, in the previously mentioned case of the letter “a” in the French language, when a tap is detected on the letter-key associated with “a” that does not follow a preceding tap on a letter-key, the first default letter-form “à” is generated, and when a tap is detected on the letter-key associated with “a” that does follow a preceding tap on a letter-key, the second default letter-form “a” is generated. In another embodiment, when a first default letter-form is generated in response to a tap on a letter-key that does not follow a preceding tap on a letter-key, and a next letter-key is tapped immediately following the tap in which a first default letter-form was generated in response, then the previously generated first default letter-form is converted to a second default letter-form in the string of text characters to be generated in response to the sequence of tapped letter-keys. In another embodiment, the default letter-form associated with a key-tap is determined with respect to the preceding letter in the sequence of tapped letters.

[0170] Thus, in accordance with one embodiment, at block 3610 if TAP_Word_Len is 0, then the process identifies a first default letter associated with the key in which the TAP location occurred. If at block 3610 TAP_Word_Len is greater than 0, then at block 3615 the process identifies a second default letter associated with the key in which the TAP location occurred.

[0171] Then at block 3620 the identified default letter is appended to the current TAP_WORD[] being formed, which at blocks 3625 and 3630 is appended to the current Word_Candidates[] list as a designated TAP word before returning at block 3635.

[0172] If at block 3603 the TAP location is not within a key associated with the generation of a letter, then at block 3640, the process determines if the TAP location occurred within the boundaries of a displayed Word Choice list, and if so, at block 3645 a Word Selection Routine 3800 is invoked to process the selection of a word or a word choice list function, and the process returns at block 3650. If at block 3640 the TAP location is not within the Word Choice list, then at block 3655 the process determines whether the

TAP location occurred within the boundaries of the BackSpace key, and if so, and if also at block 3660 TAP_Word_Len is found to be greater than 0, then at block 3665, TAP_Word_Len is decremented. If at block 3670 TAP_Word_Len is found to be still greater than 0, then processing continues as before at block 3625 to append the current TAP_WORD[] to the Word_Candidates[] list. If at block 3670 TAP_Word_Len is found to be set to 0 (indicating that any previously tapped entry has been completely deleted), then at block 3693 Word_Output is set to FALSE. If at block 3660 TAP_Word_Len is found not to be already set to 0, then at block 3690 the BackSpace function is invoked to perform its normal word processing function (i.e. delete the character to the left of the text cursor or delete the highlighted chunk of text if one exists, and so on). Following that, at block 3693, Word_Output is set to FALSE since it is no longer appropriate to assume that the user would want a space automatically output prior to a next selected word.

[0173] If at block 3655 the TAP location is not on the BackSpace key, then at block 3680 the process determines whether the TAP location occurred within the boundaries of a key associated with a function that causes the default word of a selection list to be output, and if so, at block 3685 Word_Choice is set to its default value of 0 and at block 3687 the Word Output Routine 3900 is invoked to output the default word choice to the insertion point in the text output area. If the function found at block 3680 does not cause the default word to be output, then at block 3690 the function associated with the tapped key is invoked to perform its normal word processing function. Depending on the nature of the function invoked, at block 3693 the flag Word_Output is set to FALSE if it would not be appropriate to automatically output a space prior to a next selected word (for example, following a tap on a TAB key).

[0174] In another embodiment, a special interpretation is assigned to an input path consisting of a single isolated occurrence of a DOUBLE_LETTER inflection point, where no other inflection points are detected except for the PEN_DOWN and PEN_UP location (which by definition will be detected in every input pattern) where both are detected within a maximum threshold distance along the input path from the detected center of the isolated DOUBLE_LETTER inflection point. To facilitate the rapid (and therefore, potentially inaccurate) entry of single-letter words (which would otherwise be entered by tapping the associated key and would therefore be strictly interpreted as a tap upon the letter-key within which the tap action actually occurred), an isolated DOUBLE_LETTER inflection point is also processed by the system as defining a key-board location potentially near one or more single-letter words, such that, in contrast to the processing of an input path wherein when a DOUBLE_LETTER inflection point is matched to a single letter of a multi-letter word, no skip-ping penalty is applied when a DOUBLE_LETTER inflection point is matched to a single letter of a single-letter word. This is particularly helpful in languages such as French, wherein a single letter-key may be associated with two different one-letter words (as in the "a" key, which is as-sociated with both the single-letter words "a" and "à". This approach allows both to be immediately presented in a selection list, where otherwise the only alternative for at least one of the two words is to perform a "tap-hold" on the letter-key and wait for the required time-out thresh-old to pass so that the diacritic letter-form menu is pre-sented.

[0175] Figure 3G shows a embodiment of the Update Word_Candidates[] Table Routine 3700, called at block 3385 of the of the Pattern Matching Routine 3300, and at blocks 3625 and 3630 of the of the Process TAP Inflection Point Routine 3600. At block 3705, if the current word has been flagged as not being a valid match with the current input path, then at block 3735 the routine returns without adding the current word to the Word_Candidates[] table. If not, Num_Candidates is checked to determine whether the word candidates table has yet been filled with the maximum number of valid candidates, and if not the current word is automatically added to the Word_Candidates table at 3740, and at 3745 Scoring metric is checked to determine whether the current value of Max_Metric_Value needs to be updated. If at block 3710 the Word_Candidates[] table has already been filled with a set of valid entries, then at block 3715 the value of Scoring Metric for the current word is compared to Max_Metric_Value to determine if the current word is a better match than one (or more) words currently in database. If so, the word having the highest Scoring Metric value is removed from the Word_Candidates table at 3720, and at 3725 the current word is added to the Word_Candidates[] table, sorted in descending order according to the Scoring Metric value for each entry. At 3730, Max_Metric_Value is updated to reflect the new highest value of Scoring Metric within the Word_Candidates[] table. At block 3755, control returns to the caller.

[0176] Figure 3H-1 shows an embodiment of the Word Selection Routine 3800, called at block 3645 of the Process TAP Inflection Point Routine 3600. At block 3805, the region of the Word Choice list display in which the TAP contact occurred is identified and the candidate word or list function associated with the region is determined. If at block 3810 the selected list region is not associated with a word choice, then if at block 3815 it is determined that the region is associated with the "(more)" function, then at block 3820, the process determines a next set of word candidates to display (either additional identified candidates associated with higher Scoring Metric values, or the initial set of displayed words including the default word) and at block 3825 the Display Selection List Routine 31000 is called to refresh the selection list display with the determined set of word candidates in an updated Word Choice list display. The routine then terminates at block 3830.

[0177] If at block 3815 the selected list region is not associated with the "(more)" function, then if at block 3835 it is determined that the region is associated with the "CANCEL" function, then at block 3840 the Word_Candidates[] table is cleared, Num_Candidates is set to 0, and Max_Metric_Value is set to the MAX_FLAG value, so that the Word_Candidates[] table is ready to have a new set of determined candidates added to it. Similarly, at block 3845 TAP_Word_Len is set to 0 so that a new TAP word can start to be constructed, and at block 3850, the Word Choice list display is canceled, and the display screen is refreshed to remove the previous Word Choice list display from the screen. The routine then terminates at block 3855.

[0178] If at block 3810 the selected list region is associated with a word choice, then at block 3860 the process determines whether the selected word (which is thus established as corresponding to the user's intended word) is the first word in the selection list, corresponding to the default word with a determined Scoring Metric value indicating that it is the most closely-matching word in the database. If not, then at block 3865 an Adjust Word Frequency Routine 38100 is called to determine what if any word frequency adjustments should be made in the database. If at block 3860 the selected word is the first (default) word in the selection list, then at block 3870 the Word Output

Routine 3900 is called to output the selected word to the text display area 2106 at the insertion point 2108. At block 3875, the process determines if the selected word was a constructed TAP-location word, and if so, at block 3880, if the selected word is not already in the database, then it is added to the database as a User-Defined word. The routine then terminates at block 3885.

[0179] The frequency information stored with a word in the database provides an indication of how frequently the word tends to be encountered in general usage. In another aspect, the frequency information associated with a word in a particular user's database is modified in accordance with the specific usage of that word by that user. The algorithms used to analyze the input pattern and determine which words in the database are most likely to correspond to the user's intent make use of the stored frequency information (as shown of block 3525 of Figure 3E) to optimize this determination of the most likely matching words to maximize the probability that the word determined to be the most likely match does in fact correspond to the user's intended word. Due to the fact that the system thus prioritizes words of higher frequency, there are occasions when a higher frequency word is determined to be the most likely match even when a lower frequency word actually matches the input pattern more closely. This maximizes the performance of the system, as it allows the user to enter input patterns more quickly (and consequently less precisely) while still correctly recognizing input patterns entered for frequent words when the entered input pattern happens to more closely match a lower frequency word. Since the higher frequency words are in general entered more frequently, this improves the system's performance, but also means that there will be occasions when the user's intended word is associated with a much lower frequency and as a result is determined not to be the most likely match. Furthermore, while the frequency information stored with each word in the initial database reflects the frequency of use of the word in general usage, this does not mean that the frequency with which a particular user enters a word will always correspond to its frequency in general usage. In addition, the actual performance of the system will always be a function of the particular idiosyncrasies of how a user traces out input patterns. In addition, in order for the user's experience of the system to be relatively consistent, it is preferable to keep the database relatively stable, and avoid having pairs of words frequently “trade places” in the system's determination of which is the more likely word.

[0180] In another aspect, when the user selects as the word to be output a word other than the default word (the word determined by the system to be the most likely matching word), then the frequency associated in the database with the lower frequency word is increased. In another aspect, the frequency associated with the higher frequency word is increased. In another aspect, the frequency associated with a word in the database is increased only when the system has determined that a higher frequency word is a more likely word than a lower frequency word, and where the user selects the lower frequency word as the word to be output, then. In another aspect, when the system determines that a higher frequency word is a more likely word than a lower frequency word, and where the user selects the lower frequency word as the word to be output, then the frequency associated in the database with the lower frequency word is increased only when the lower frequency word actually matches the input pattern more closely.

[0181] {HERE} In another aspect, in order to preserve the efficient operation of the system, special treatment is accorded to “high-frequency words” that are associated in the database with a frequency higher than a determined threshold. In one aspect, when the system determines that a lower frequency word is a more likely word than a high-frequency word, and where the user selects the high-frequency word as the word to be output, then the frequency associated in the database with the lower frequency word is further decreased. In another aspect, when the system determines that a high-frequency word is a more likely word than a lower frequency word, and where the user selects the lower frequency word as the word to be output, then the frequency associated in the database with the lower frequency word is increased only when the frequency associated with the lower frequency word is less than a determined threshold frequency. The usage of high-frequency words tends to be relatively consistent across users, such that it would tend to be detrimental to system performance to allow a lower frequency word that has an input pattern that is similar to that of a high-frequency word to also be assigned a frequency that is close to that of the high-frequency word. In another aspect, when there are scoring penalties assessed against the selected word and these penalties exceed a determined maximum threshold, then no adjustments are made to word frequencies.

[0182] In accordance with certain of the above-mentioned aspects, Figure 3H-2 shows an embodiment of an Adjust Word Frequency Routine 38100. In accordance with one aspect, at block 38105 the process determines whether any inflection point- or letter-skipping penalties have been assessed against the selected word, and if so, proceeds to block 38130 and flags that no change will be made in word frequencies in the database as a result of the current selection. The process then returns at 38135. If at block 38105 no skipping penalties have been assessed, then in another aspect, at block 38110, the process determines if the frequency associated with the default word in the selection list is greater than the frequency associated with the selected word. If so, then in another aspect, at block 38115, the process determines if the average distance of the letter-keys of the selected word from the corresponding input path points with which they are matched (as determined at block 3505 of Figure 3E) is less than the corresponding average distance determined for the default word in the selection list. If not, then the process again proceeds to block 38130 and returns without changing any word frequencies in the database, but if so, then in another aspect, at block 38120, the process determines if the frequency associated with the selected word is less than a determined frequency promotion threshold level. If not, then in another aspect at block 38125, the process determines if the frequency associated with the default word is greater than a determined high-frequency word threshold level. If not, then at block 38140 the frequency associated with the default word is decreased before returning at block 38145, but if so, the process again proceeds to block 38130 and returns without changing any word frequencies in the database. If at block 38120 the process determines that the frequency associated with the selected word is less than a determined frequency promotion threshold level, then at block 38170 the frequency associated with the selected word is increased before returning at block 38175.

[0183] In another aspect (not shown in Figure), if at block 38110, the process determines if the frequency associated with the default word in the selection list is not greater than the frequency associated with the selected word, then the process returns without changing any word frequencies in the database. In the aspect shown in Figure 3H-2, the process proceeds to block 38150 and determines if the frequency associated with the selected word is greater than a determined high-frequency word threshold level. If so, then at block 38155 the frequency associated with the default word is decreased before returning at block 38160. If the selected word is a high-frequency word, then at block 38165, the process determines if the frequency associated with the selected word is less than a determined frequency promotion threshold level. If not, then again at block 38155 the frequency associated with the default word is decreased before returning at block 38160. If at block 38165, the process determines that the frequency associated with the selected word is greater than a determined frequency promotion threshold level, then at block 38170 the frequency associated with the selected word is increased before returning at block 38175.

[0184] In another aspect, when there are one or more words in addition to the default word that are also assessed a lower Scoring Metric value than the selected word, then the analysis shown in Figure 3H-2 is performed for all such words that are determined to be a more likely match with the input path by virtue of receiving a lower Scoring Metric value than the selected word.

[0185] Figure 3I shows an embodiment of the Word Output Routine 3900, called from block 3230 of the Input Pattern Analysis Routine 3200, and at block 3687 of the Process TAP Inflection Point Routine 3600, and at block 3820 of the Word Selection Routine 3800. At block 3905, if no valid word candidate was identified as potentially matching the current input path, then execution proceeds to block 3925 to clear the Word_Candidates[] table as described below. Otherwise, in one embodiment, if more than one valid candidate was identified in the current selection list as a potential match for the current input path, then at block 3907 the process records the identity of the word to be output, associated with the identities of one or more the most-closely-matching candidates. This information is used to support an embodiment of a "Re-Edit" function, whereby when the user observes that a previously output word does not correspond to the user's intended word, the output word is selected and a Re-Edit function is invoked which causes the process to search its record of recently output words, and if the selected word is found, then a selection list is re-constructed consisting of the words whose identities were recorded as associated with the output word. Since the user's intended word tends to occur somewhere in the selection list of alternate word choices in the overwhelming majority of cases in which there have not been gross errors in entering the input path, this embodiment greatly simplifies the task of correcting such mistakes where the default word is unwittingly accepted for output. This Re-Edit function makes it easy to correct such words, since selecting the intended word from the displayed Re-Edit selection list automatically replaces the previously output word in the generated text.

[0186] Returning to Figure 3I, at block 3910, if the automatic spacing flag Word_Output is set to TRUE, then a space is output to the text insertion point at block 3915 prior to outputting the selected word candidate at block 3920. Also, Word_Output is set to TRUE so that if a next word is subsequently output, a space will be automatically generated prior to the next word. Then at block 3925, where execution also resumes if the current Word_Candidates[] table is found to be empty at block 3905, the Word_Candidates[] table is cleared, Num_Candidates is set to 0, and Max_Metric_Flag is set to the MAX_FLAG value, so that the Word_Candidates[] table is ready to have a new set of determined candidates added to it. Then, at block 3930 TAP_Word_Len is set to 0 so that a new TAP word can start to be constructed, and at block 3935, the Word Choice list display is canceled, and the display screen is refreshed to remove the previous Word Choice list display from the screen (if one was displayed). The routine 3900 then terminates at block 3940.

[0187] Figure 3J shows a embodiment of the Display Selection List Routine 31000, called from block 3125 of the Main Processing Routine 3100, and from block 3825 of the Word Selection Routine 3800.

[0188] At block 31005, if a user preference has been set to always display a word choice selection list, then the process immediately continues from block 31035 to display the selection list as described below. Otherwise, at block 31010 the process determines if no valid matching candidates have been identified, and if so, at block 31031 if the user has set a preference for any visual or auditory signal to be generated when no valid match is identified for an input pattern, then the corresponding signal is generated before returning at block 31033. If at block 31010 at least one valid matching candidate has been identified, then at block 31015 the process determines if more than one valid matching candidate has been identified, and if not, at block 31025 Word_Choice is set to 0 so that the following call at block 31030 to the Word Output Routine of Figure 3I causes the default most-closely matching word to be generated as text output. If more than one valid matching candidate has been identified at block 31015, the process proceeds to block 31020.

[0189] In another aspect, when the system determines that the likelihood that the default choice is in fact the user's intended word is sufficiently greater than the likelihood of the second-most-likely word, then the default word is output without displaying a list of alternate word choices. In another embodiment, at block 31020, when the difference between the Scoring Metric values for the first and second words falls below a user-selected threshold, the process proceeds to block 31025 as described above to output the default word. In another aspect, the ratio between the Scoring Metric values for the first and second words is compared to a user-selected threshold.

[0190] In another aspect, when a list of possible word choices is presented to the user for selection of the intended word, the list is sorted according to a primary scoring metric, with the words sorted from the most-closely matching word to the least-closely matching word. In another aspect, the most-closely matching word is presented in a distinctive visual fashion as a function of the magnitude of the difference between it and the second-most-closely matching word.

[0191] In another aspect, when the difference in likelihood of the two most likely word choices falls below one or more determined thresholds, the user is informed through an auditory and/or visual display so that the user can choose to divert their attention to attend to the word that has just been output (or that will be output if the user performs any action other than selecting an alternate word choice from the list) by the system only when there is some reasonable likelihood that the user may in fact need to select a word other than the default choice. This is a significant benefit as it allows the user to more effectively direct their attention while inputting text. For example, in one embodiment, at block 31065, the outline of the keyboard (on which the user's attention is generally focused) is displayed in a color which varies according to the threshold range into which the difference in likelihood falls. In another embodiment, a distinctive tone is generated which varies according to the threshold range into which the difference in likelihood falls.

[0192] Figure 6 shows an example of a virtual keyboard 2104 that is displayed in accordance with certain of the above embodiments. The default word choice “track” 6000 has already been output at the text insertion point 6002 without displaying a list of alternate word choices, in accordance with the embodiment wherein the system has determined that the likelihood that the default choice is in fact the user's intended word is sufficiently greater than the likelihood of the second-most-likely word. However, in the example shown in Figure 6, while large enough to exceed the threshold determining that a selection list will not be displayed, the difference in likelihood of the two most likely word choices falls below one or more determined thresholds wherein a visual display is created to notify the user that the difference in likelihood falls below a determined threshold. In accordance with one embodiment, a visual highlight 6004 has been added to the outline of the keyboard 2104, where the color of the outline corresponds to threshold range encompassing the difference in likelihood between the default word “track” 6000 and the second-most-likely word (“real” in the example depicted in Figure 6).

[0193] In another embodiment, more than one threshold is used to determine how close the scoring metric scores of one or more of the next-most-closely-matching words are to the most-closely-matching (first) word, and an auditory and/or visual signal is generated to give some indication to the user how close a match each word is to the first. In another aspect, each word following the most-closely matching word is presented in a distinctive visual fashion as a function of into which difference threshold the difference between it and the most-closely matching word falls, such that words whose scoring metric differences (or, in another embodiment, ratios) fall between the same thresholds form a threshold difference group. At block 31035, the display flags used to indicate any distinctive visual highlighting to be applied to each word in the list are cleared. Then, at block 31037, the process identifies any candidate words for which the ratio of the primary Scoring Metric value for the word (shown in Figure 3E at block 3525 as *adjustedScore*) to that of the first word falls below one or more determined threshold values. Then at block 31040, a display flag is set for each word identified at block 31935 corresponding to the lowest threshold criterion met by the ratio of the word's Scoring Metric value to that of the first word.

[0194] In another embodiment, more than one type of scoring metric calculation is used to determine how closely each analyzed word matches the input pattern, and the two most-closely-matching words are compared with respect to two or more distinct scoring metric calculations to determine whether the difference between the two words fails to exceed one or more thresholds. For example, one embodiment of a secondary scoring metric is shown in Figure 3E at block 3505 where the value avDistance is calculated for each candidate word. In another embodiment, when more than one type of scoring metric calculation is used to determine how closely each analyzed word matches the input pattern, corresponding thresholds are used for each scoring metric to define threshold difference groups for each scoring metric used, and the list of possible word choices is presented to the user such that all words that fall into the same threshold difference group (as defined by one or more of the scoring metrics used) are presented in a contiguous group in the list, and wherein all words of a given difference threshold group are presented in the same distinctive visual fashion (for example, with a specific color background on the display).

In accordance with these embodiments, at block 31045 the process identifies any candidate words for which the ratio of the secondary Scoring Metric value for the word to that of the first word falls below one or more determined threshold values. Then at block 31050, a display flag is set for each word identified at block 31945 corresponding to the lowest threshold criterion met by the ratio of the word's secondary Scoring Metric value to that of the first word. Then at block 31055, the list of candidate words is re-sorted in order to move any words in the list for which the secondary display flag has been set up to a position in the list immediately following the last word in the same threshold group (which therefore has the corresponding primary display flag set). If no word has been identified as belonging in the same threshold group according to the primary scoring metric, then the words in a given sec-ondary threshold group are moved to follow words in any higher precedence threshold group, or if none exist, then to immediately follow the first word. Finally, at block 31060 a Word Choice list is created and displayed at the text insertion point showing the Num_Display Word_Candidates[] entries starting with the first word, where any words identified as belonging to the same threshold group are visually indicated in a distinctive manner.

[0195] Figures 7A and 7B shows an example of a word selection list that is modified and displayed in accordance with certain of the above embodiments. Figure 7A shows an input path 7000 entered by a user where the intended word was the word “three.” The user enters an input path that begins at a PEN_DOWN location 7002, proceeds in sequence to the vicinities of the letter-keys associated with “h” and “r”, and then attempts to enter a DOUBLE_LETTER gesture 7004 in the vicinity of the letter-key associated with “e” but fails to complete a recognizable gesture. The selection list 7006 shown in Figure 7A shows the list that would result in accordance with the various aspects of the present invention but without implementing the above-described embodiment in which all words that fall into the same threshold difference group (as defined by one or more of the scoring metrics used) are presented in a contiguous group in the list. Due to the unrecognized attempted DOUBLE_LETTER gesture at 7004, the primary Scoring Metric associated with the intended word “three” is not good enough for the word to appear on the displayed selection list 7006 of the top four candidates, since Scoring

Metric "three" ranks seventh among the matched candidate words with respect to the primary Scoring Metric. As shown in Figure 7A, "thread" and "enter" are the two top-ranked candidate words, both of which appear in the same highlighted fashion 7008 associated with the lowest threshold difference group, including those word candidates whose Scoring Metric most closely matches that of the top candidate (and thus, in the example shown, including the top candidate itself). The third-ranked candidate "threat" is shown is a second distinct highlighted fashion 7010 associated with the next lowest threshold difference group, and the fourth-ranked candidate tuner fails to meet any difference threshold and appears without distinctive visual highlighting 7012.

[0196] Figure 7B shows the same input path 7000 with the selection list 7014 that results in accordance with the various aspects of the present invention and also including the above-described embodiment in which all words that fall into the same threshold difference group (as defined by one or more of the scoring metrics used) are presented in a contiguous group in the list. Since the intended word "three" is also included in the lowest threshold difference group with respect to the secondary Scoring Metric (*avDis-*

tance, as shown in Figure 3E at block 3505), in Figure 7B the intended word “three” appears third in the displayed selection list 7014, immediately following the two top-ranked candidate words “thread” and “enter,” and appearing in the same highlighted fashion 7008 associated with the lowest threshold difference group. This aspect thus makes it possible for the user to immediately find and select the intended word from the first displayed selection list.

[0197] While the embodiment of the invention has been illustrated and described, as noted above, many changes can be made without departing from the spirit and scope of the invention. Accordingly, the scope of the invention is not limited by the disclosure of the embodiment.